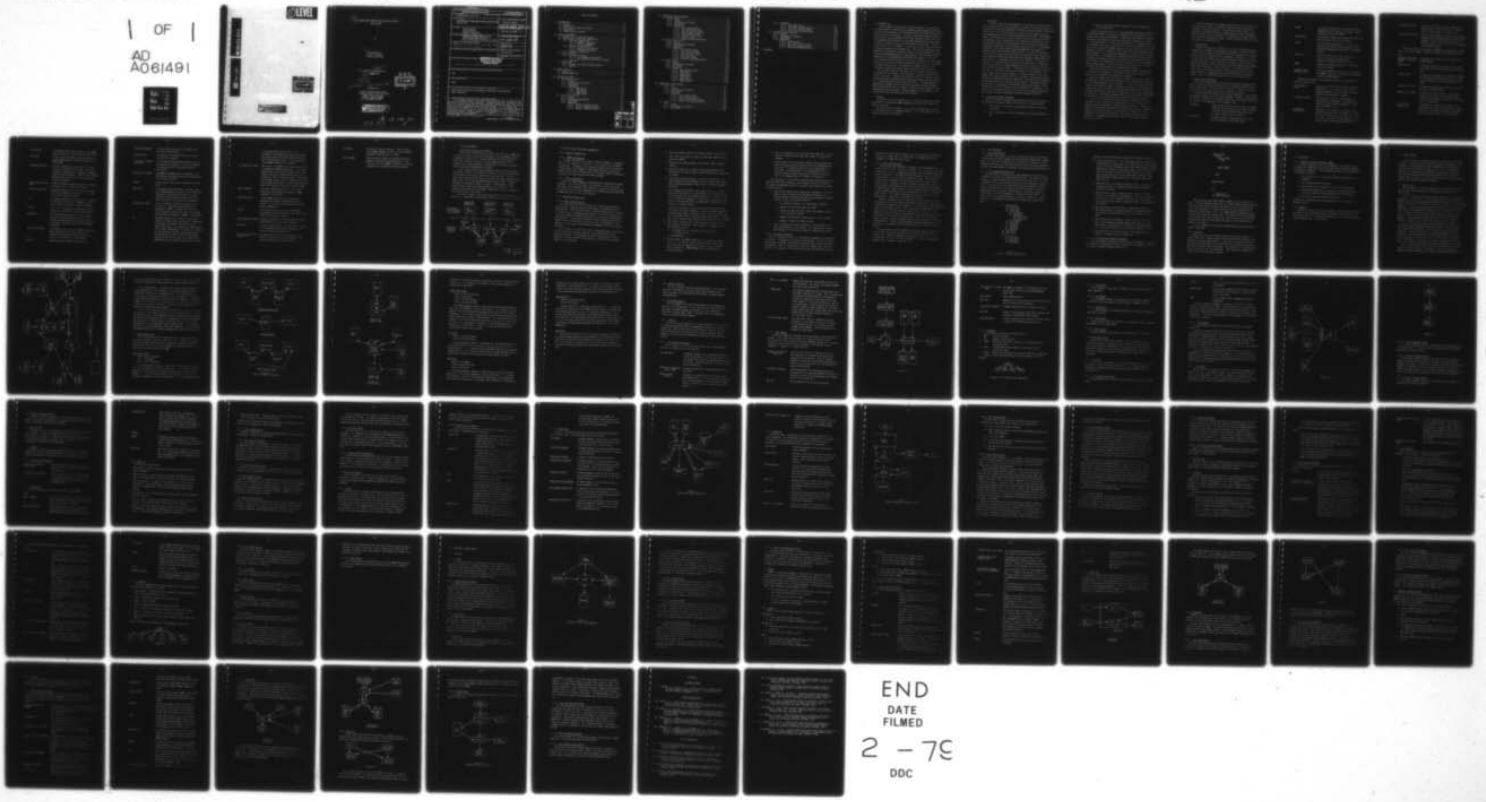
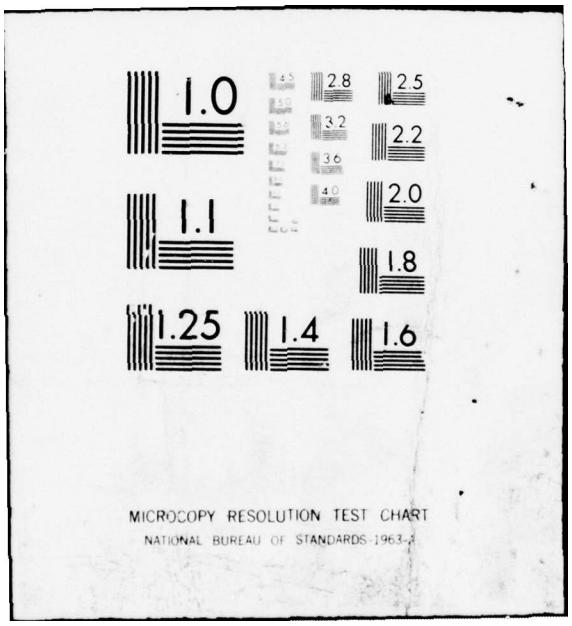
AD-A061 491 MICHIGAN UNIV ANN ARBOR GRADUATE SCHOOL OF BUSINESS--ETC F/G 9/2
MICHIGAN DATA TRANSLATOR DESIGN SPECIFICATIONS. VERSION IIB.(U)
OCT 77 D DESMITH, L A HUTCHINS, M STOLARCHUK DCA100-75-C-0064
UNCLASSIFIED WP-77-DT-3.8 SBIE-AD-E100 111 NL

1 OF 1
AD A061491



END
DATE FILMED
2 - 79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL

AD A061491

DDC FILE COPY

DDC
RECEIVED
NOV 22 1978
B

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

(6) MICHIGAN DATA TRANSLATOR DESIGN SPECIFICATIONS.
Version IIB.

by

(10) Donald DeSmith,
Linda A. Hutchins
Michael Stolarchuk

(9) (14) Working Paper WP-77-DT-3.8
(11) October 1977
(12) 76p.
(18) SBIE (19) AD-E100 111

DDC
RECEIVED
NOV 22 1978
B

Prepared for

Defense Communications Agency
Command & Control Technical Center
WWMCCS ADP Directorate
Reston, Virginia 22090
(15) DCA 100-75-C-0064

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

78 11 09 036
409 349 LB

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Working Paper 77 DT 3.8	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) MICHIGAN DATA TRANSLATOR DESIGN SPECIFICATIONS Version IIB		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER WORKING PAPER 77 DT 3.8
7. AUTHOR(s) Donald DeSmith Linda Hutchins Michael Stolarchuk		8. CONTRACT OR GRANT NUMBER(s) DCA 100-75-C-0064 new
9. PERFORMING ORGANIZATION NAME AND ADDRESS Database Systems Research Group 276 Business Administration Univ. of Michigan, Ann Arbor, MI 48109		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 32017K, 27400, 27402
11. CONTROLLING OFFICE NAME AND ADDRESS DEFENSE COMMUNICATIONS AGENCY, C422 WASHINGTON D.C. 20305		12. REPORT DATE October 1977 13. NUMBER OF PAGES 75
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

CLEARED FOR OPEN PUBLICATION, DISTRIBUTION UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

N/A

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Design Specifications, IDS databases, Conversion, Restructuring
Data Translation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document describes the capabilities and design specifications of the Version IIB, Release 1 Data Translator. Although the previous versions of the Data Translator were designed to demonstrate the research performed, this version has as its design objective usability and efficiency. It incorporates the experience and capabilities of the previous versions of the Translator developed during the first four years of research and development for the Defense Communications Agency. The main emphasis of this version is on the conversion and restructuring of IDS databases. See also AD's A061 498 and A061 455.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

PER FORM 50
A

1.0 INTRODUCTION

The development of the Version IIB Data Translator is a cumulation of several years of research by the Database Systems Research Group (nee Data Translation Project) of the Graduate School of Business Administration at the University of Michigan. The costly problem of database conversion and the lack of any technological tools was recognized several years ago by the WWMCCS ADP Directorate of the Defense Communications Agency which initiated a research project with the University of Michigan. The goal of this research is to automate, in so far as possible, the costly process of database conversion and restructuring. The research performed by the Database Systems Research Group has resulted in the development of a data translation methodology which is based upon high level data description languages and the implementation of generalized software. The high-level languages provide the basis for the user to describe the data inputs and outputs and the transformation required to perform the data conversion and restructuring process. The generalized software, termed a Data Translator, performs the process being directed by these languages.

This document describes the capabilities and design specifications of the Version IIB, Release 1 Data Translator, hereafter referred to as the Data Translator. Although the previous versions of the Data Translator were designed to demonstrate the research performed, this version has as its design objective usability and efficiency. It incorporates the experience and capabilities of the previous versions of the Translator developed during the first four years of research and development for the Defense Communications Agency. The main emphasis of this version is on the conversion and restructuring of IDS databases.

1.1 Purpose

The purpose of the Data Translator is to provide a generalized, automated means of reorganizing WWDMS databases by altering their logical and physical structure.

Any combination of Sequential, ISP, and IDS databases can be translated into any collection of WWDMS database(s). Up to five databases can be input and created in the process.

1.2 Background

Translator research and development at the University of Michigan has led to the design of three experimental translators (a Prototype, Version I, and Version II) and two operational translators (Version IIA and Version IIB). Research for each version focused on a specific set of capabilities and the implementation served to demonstrate these capabilities. Each experimental translator was built upon the capabilities of the previous version and produced a major contribution to the generalized translation process. The Prototype translator was developed to test the design hypothesis and the general paradigm for the translation process [UT1,2,3]. It performed conversions from a NIPS variable blocked sequential file (IBM) to a Honeywell 6000 System Standard Format database. The major contributions of the Prototype translator design were the validation of the approach and establishment of common interfaces among the Translator modules which has been beneficial in subsequent versions of the Translator. The Version I Translator incorporated a basic restructuring capability of "Compression" which was necessary to transform WWDMS hierarchical structures into the two-level NIPS structure [UT5,6]. The emphasis of the Version II Data Translator was on hierarchical restructuring capabilities and extended input/output capabilities to handle sequential, indexed sequential (ISP), and network (IDS) data structures [UT7,R2]. The Version IIA Translator design focused on expanded restructuring capabilities for network (IDS) databases and increased practical usability [UT8,9,10,UR3,5]. Subsequent releases of the Version IIA focused on the operation and usability aspects of the Translator [UR6,7,UT11] (hence, the term "operational" translator). The Version IIB Translator maintains this focus and has developed capabilities for increased efficiency throughout the Data Translator [UT12,13].

1.3 The Version IIB Data Translator

With a view toward operational effectiveness, testing of the Version IIA Translator uncovered a number of inefficiencies and complications.

1. Inefficiency in the Reader and Restructurer modules of the Data Translator,
2. Lack of generality in the Writer module of the Data Translator, and

3. Complexity and multitude of steps which must be performed by the Translator user.

After many exhaustive tests, it was determined that the time required by the Reader module to access source databases was proportional to the square of the size of the database. A new reader algorithm for increasing the speed of the Reader size has been designed to mitigate this problem.

The performance of the Version IIA Restructurer is inadequate and its user interface (the user written TDL description) is cumbersome. Even when only a small portion of database needs to be translated, the Version IIA Translator requires a complete TDL description and actually processes the entire database. This encourages unnecessary TDL errors and slow Translator performance. To alleviate this problem, the Version IIB Data Translator design incorporates the partial restructuring facility which allows a user to specify TDL descriptions for only that portion of the database which must be restructured. Using this facility, the Restructurer then needs to process only that portion of the database which changes.

In the Version IIA Translator, the allowable target database format is limited to IDS databases. The Version IIB Translator extends this choice to include both ISP and Sequential databases.

The user interface with the Version IIA Data Translator was judged to be cumbersome in that errors are frequent and are difficult to detect until well into the execution of the Translator. To simplify the user interface, a control card generator,added to the Data Translator in the Version IIB, relieves the user of the task of modifying the prototype control cards for a translation entirely on his own. A monitor facility is also incorporated to aid the user in determining whether the translation process is executing, and to give the user some idea of how long the process will continue.

Even with the increased efficiency inherent in the Version IIB Data Translator design, there will be translations which are so large as to require execution time beyond typical resource constraints. While this problem is addressed by the Version IIA translator by allowing independent execution of translator modules, the Version IIB allows an even finer breakdown of the translator execution. Each module is able to run incrementally, i.e., executed in many small blocks of time rather than one large block of time.

With the addition of incremental processing, partial restructuring, and an efficient reader, the computer resources required by the Data Translator are greatly reduced. The execution monitor, control card generator, and extended choice of target database formats increase the Data Translator's user friendliness. The combination of these improvements provides the potential Version IIB Data Translator user with a tool far superior to the Version IIA.

1.4 Organization

Section 1.5 gives brief definitions of some of the terminology and concepts used throughout the remainder of the paper. Section 2.1 gives a high level description of the Data Translator organization and function while its specific capabilities are enumerated in Section 2.2. Section 2.3 specifies the hardware and software environments necessary for use of the Data Translator. A more detailed description of the relationships among the modules of the Data Translator, and the databases used internal to the Data Translator, is given in Section 3. The Data Translator itself can be broken down into eight separate modules which fall into three categories: language analyzers, execution modules, and support modules. These are described in Sections 4,5, and 6, respectively.

1.5 Terminology and Concepts

This section briefly defines some of the terms used throughout this paper. General concepts are discussed first, followed by definitions of more specific terms. This section may be skipped or used only for occasional reference by those familiar with translation concepts.

The Data Translator consists of two language analyzers, three translator execution modules, and three support modules. These are, respectively, the IDS and TDL Analyzers; the Reader, Restructurer, and Writer; the Front End, ADBMS, and the Aggregate Schema Processor, ASP.

IDS Analyzer	A Data Translator module that accepts an extended IDS MD section, which describes a user's database in textual form,and produces SDDL tables which are used by other translator modules.
TDL Analyzer	A language analyzer that accepts as input TDL text and produces as output TDL tables.

Reader	The Translator module which transforms the user's database(s) into a logically equivalent internal database processable by ADBMS.
Restructurer	The Translator module that produces the relational RIF from source RIF and the TDL tables.
Writer	The final translation module. Its job is to transfer data records from the target RIF into the target database(s), preserving all information content.
Front End	The pre-processor to the Data Translator; it constructs control card images and creates all necessary translation files.
ADBMS	A DBTG-like database management system used by the Data Translator.
Aggregate Schema Processor (ASP)	A database management system used in conjunction with ADBMS to facilitate integration and accessing of multiple ADBMS databases.

Definitions of general terms used with respect to the operation of the Data Translator are:

Direct Reader to Writer	A mode of the translator that does not require Restructuring. It is used to change physical database parameters when the logical structure does not vary from source to target.
Incremental Reading	The process of reading a source database in multiple runs of the Reader. The Reader can be used to process the source database in a user-specified number of runs.
Incremental Restructuring	The process of restructuring a database using several Restructurer runs as opposed to a single run. Each run processes only a few access paths at a time. The target records produced by each run are stored in the same relational RIF.

Incremental Writing	The process of writing a target database over multiple runs. Each run consists of creating specific record types to the target file.
Partial Restructuring	A mode of the Translator in which the Restructurer processes only the changing portion of the source database. Used primarily when the majority of the source data is static, i.e., will be unchanged in the target database.

Other terms which are referenced throughout this document are defined below. Terms which appear in specific sections are defined as they are used.

Aggregate Schema Data Definition Language (ASDDL)	A language used to describe the integration of ADBMS databases.
ASDDL Analyzer	A language analyzer that accepts as input ASDDL and outputs an initialized Aggregate Schema database.
Bachman Diagram	A figure which represents the schema of a database. Record types are represented by boxes; set types are represented by lines connecting record type boxes.
Database Initializer	A program that initializes a file so that it can be used as an ADBMS database.
Database Tables (DBT)	The output of the DDL Analyzer. These tables contain a description of the database schema in a form usable by ADBMS and are stored in the first pages of the database.
Data Definition Language (DDL)	A language used to describe the schema of an ADBMS database in terms of the three basic ADBMS constructs: the item, the record, and the set. DDL text serves as input to the DDL Analyzer.

DDL Analyzer	A language analyzer that accepts as input ADBMS DDL and produces as output Database Tables (DBT).
DDL Writer	A module that scans SDDL tables and produces as output ADBMS DDL text suitable for input to the DDL Analyzer.
Extended MD Section	The user's databases are described to the Data Translator through IDS MD sections. These MD sections are augmented with additional information required by the Translator. There are two extended IDS MD sections, one for the source and one for the target databases.
Integrated Data Store (IDS)	A network database management system on the Honeywell 6000 used as a source of translation.
IDS Query Dictionary	A database produced from the extended MD section by the IDS Translator in query mode. It is read in as input to the IDS Analyzer.
ISP	An indexed-sequential file organization. Databases in ISP form (if conforming to WWDMS T-2 rules) can be read and created by the Data Translator.
Library	A file used to hold the object modules of several logically or functionally related subroutines.
MD Section	The IDS MD section used by application programs. It begins with "MD database name" and differs from the extended MD section in that no 61 levels are present. Used and compiled into the Reader and Writer.
Primary Key (ADBMS)	A collection of data items in an ADBMS hash record whose combined values uniquely determine an instance of the hash record type which can be used in the storage and retrieval process.
Relation	Synonymous with set (ADBMS) and chain (IDS).

Restructured Record	A target record type which is different from its source record type(s).
Restructuring	The process of altering the logical structure or schema of a database.
Restructurer Internal Form (RIF)	The standardized-format ADBMS database on which the Restructurer operates. All user's databases are converted to this form before they are restructured.
Relational RIF (RRIF)	The ADBMS RIF database that is produced by the Restructurer from the source RIF and the specification in the TDL tables.
Schema	A description of the logical structure of a database.
SDDL Tables	The output of the IDS Analyzer. The SDDL tables are an ADBMS database in which all the information contained in the extended MD section is stored. There are two SDDL table databases; one for the source and one for the target user's databases.
Sequential Database	Any user database not under the control of ISP or IDS, e.g., a system standard format file, manageable by GFREC. The Data Translator will accept sequential databases as input to the Reader provided that they conform to WWDMS T-2 restrictions.
Set	An ADBMS construct used to define a relationship between two record types, one of which is designated as the owner and the other designated as the member record type. An instance of an ADBMS ordered set is always implemented as a linked list consisting of one instance of the owner record type for that set type and one or more instances of the member record type for that set type. An instance of an ADBMS match-key set always consists of one instance

	of the owner record type and one or more instances of the member record type, each having the primary key of the owner record instance in their set-significant items. The IDS equivalent of the ADBMS ordered set is the chain.
Set Significant Items	Those items in an ADBMS record type which are used to implement an ADBMS match-key set. For a given match-key set, set significant items exist in the member record and correspond one-to-one to the primary key items in the owner record. An instance of the member record type is a member of the match-key set when the values of its set significant items match the primary key items of an owner instance.
Source Database	The user's database which will be translated. Three types of databases are accepted: WWDMS sequential, ISP, and IMS.
Source RIF (SRIF)	An RIF (ADBMS database) which is produced by the Reader and used by the Restructurer and Writer. It is logically equivalent to the user's source database(s).
Subfile	A file which contains part of an IDS database. IDS databases can be divided into many GCOS physical files for a variety of reasons.
System Generation Tape	A tape which contains all of the files and control card necessary to run the Data Translator.
Target RIF	A logical concept of how the Writer views its input. It is comprised of two physical databases - the source RIF and the relational RIF.
Translation Definition Language (TDL)	A language used to describe the mapping of the logical structure of the source database to the logical structure of the target database.

TDL Tables	The output of the TDL Analyzer. The TDL tables are an ADBMS database ir which all the information contained in the TDL text is stored.
Work Database	There are two of these, one produced during the Reader, and one during the Restructurer. Used to assist in writing ADBMS or ASP DDL for internal translator use. See DDL Writer Work Database.

2.0 DESIGN REQUIREMENTS

2.1 Description of System and Function

The Version IIB Data Translator makes it possible for a WWDMS user to alter the physical and logical structure of existing WWDMS databases (sequential, ISP, and IDS) to obtain completely new valid sequential, ISP, or IDS database structures. Multiple WWDMS databases can be combined into a single database or restructured into multiple databases. No user application programming is required.

User's descriptions provide parameters which drive a generalized translation algorithm. As indicated in Figure 2-1 the translation process can be viewed as two phases: specification and execution. The user inputs specifications of the source (original) and target (desired) databases as well as the required transformation between them. Language analyzers are executed to compile the user's specifications. These specifications control the execution phase of the Translator. The execution phase consists of the Reader, Restructurer, and Writer modules, which are invoked in that order.

The user interface to the Data Translator will be discussed in Section 2.3. More detail concerning the design will be given in Section 3.

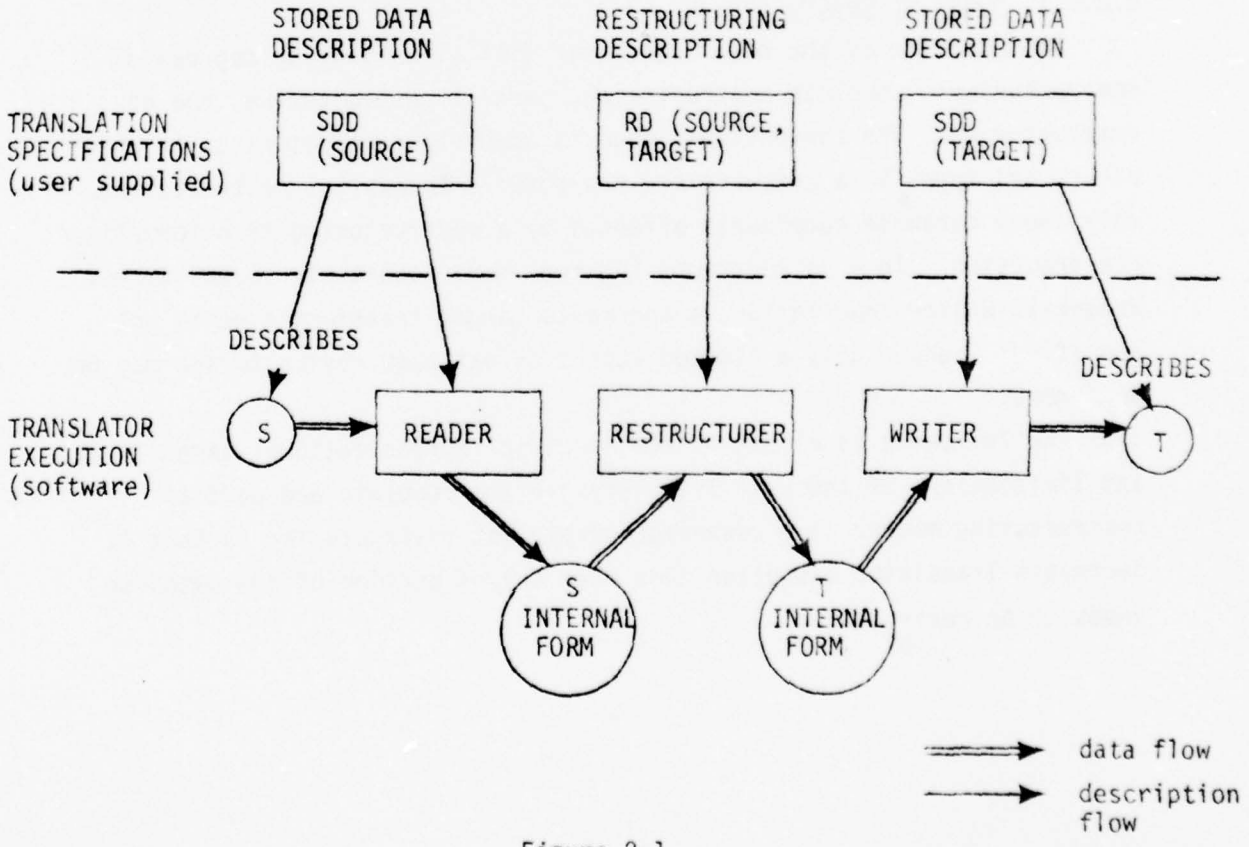


Figure 2-1

2.2 Description of Translator Capabilities

2.2.1 Technical Capabilities

2.2.1.1 WWDMS T-2 Databases

The Data Translator will accept as input any WWDMS T-2 sequential, ISP, or IDS databases. In addition, it will accept most non-WWDMS IDS databases. Non-WWDMS ISP or sequential databases will be accepted if they conform to the WWDMS T-2 requirement that every record must contain a field with a constant value (of no greater than 12 characters) to identify the record type of the record instance.

2.2.1.2 Multiple Databases

Up to five source databases, representing any combination of WWDMS sequential, ISP, or IDS databases, may be handled in one translation. From these source databases, the Translator is capable of generating (one at a time) up to five target databases, also representing any combination of WWDMS sequential, ISP, or IDS databases.

2.2.2 Operational Capabilities

2.2.2.1 Modes of Restructuring

This version of the Data Translator will allow varying degrees of restructuring: complete restructuring, partial restructuring, and no restructuring. The conventional mode is complete restructuring, in which all record types in a database are processed. In partial restructuring, only those database components affected by a restructuring transformation are processed. In a no restructuring run, more accurately termed direct Reader-to-Writer translation, a source to target transformation is not specified. Hence, only a limited amount of database restructuring may be performed.

The following is a list of the specific reorganizational capabilities and limitiations of the Data Translator in the complete and partial restructuring modes. The advantage of partial restructuring is that it decreases Translator execution time when only a portion of the database needs to be restructured.

1. Relations between records can be added, deleted, or modified.
2. Data items can be added, deleted, or have their physical representation changed.
3. Records can be added, deleted, or have their items' contents changed.
4. New record types can be created in the target database from aggregations of source records. Duplicate data can be eliminated or created.
5. Contained-in-repeating groups at the 03 thru 49 level in IDS source databases can be expanded into their own 01 level record type to become the detail of the 01 record in which they were contained.
6. User-implemented relations, such as phantom pointers, pointer arrays, and match-keys within and between source databases, must be restructured into legal WWDMS sequential, ISP, and IDS relations in the target databases. These features are not under the control of IDS and therefore are undesirable. They will not be allowed in a target database.
7. The physical structure of the target database may be different from that of the source database. Logically deleted records may be removed and changes in page size, page range, place near, etc. will be allowed in order to improve database performance. Dynamic page ranges cannot be described, and therefore are not translatable.
8. IDS primary records at the system level which contain no data item cannot be translated. The following is an example of an untranslatable record.

```
01  RECORD-NAME TYPE IS 1 RETRIEVAL VIA P-FIELD FIELD.
    02  P-FIELD PIC 9(8)
98  CHAIN MASTERS. . . . .
```

9. Packed decimal and COMPUTATIONAL data types cannot be handled directly. If a source database contains either of these data types they must be redescribed as character types to insure proper translation. COMPUTATIONAL-1 and -2 data types are handled correctly.

10. Due to the complexities of the translation algorithm, it is not possible to perfectly duplicate chain orders in a target IDS database.

11. Phantom chains are not allowed in either source or target databases since there are many possibilities for implementing them and there is no guarantee that the list has been properly maintained. (A phantom chain is an implementation of a 1:n relation between a group A and several instances of group B. This is an implementation of a linked list structure.)

The following is a list of the reorganizational capabilities of the Data Translator when performing a direct Reader-to-Writer translation. This mode is advantageous because it requires fewer resources to produce a target database.

1. Certain IDS databases can be physically reformatted. If an IDS database does not contain user-implemented relations (i.e., phantom pointers, match-key relations, and contained-in-repeating groups), it may be translated, enabling:

   a. Re-CALCing of records (e.g., page ranges, intervals).

   b. Addition of new record types to be CALC.

   c. Changes in page size, page ranges, area numbers, percent fill, and subfile contents.

   d. Changes in PLACE NEAR, authority keys, or chain ordering.

   e. Removal of logically deleted records.

2. Identity conversion of source sequential or ISP databases into a target IDS database will be allowed. This means that the target IDS database will have the same logical structure as the source.

2.2.2.2 Incremental Execution

One of the problems inherent in translating a database or a combination of databases is the amount of processor time required. In order to handle this problem the Reader, Restructurer, and Writer modules may be executed incrementally, that is, in a series of short runs rather than in one very long run. The impacts of this feature are to make the translation process

possible even when computer resources are constrained,and to reduce the effect of a system crash on a translation. Incremental execution is allowed in all modes of restructuring.

2.2.3 Timing and Flexibility

The execution time required by the Data Translator depends on a number of factors and is difficult to predict. It is useful to break down the Translator into its components and consider the time required for each module individually. The execution time required for each run of the IDS or TDL Analyzer is on the order of tenths of an hour. Ten to fifteen minutes should be adequate to analyze all but the most complex user specifications. Execution time for the Reader, Restructurer, and Writer varies linearly with respect to the size of the database, assuming that all other factors remain constant. For the translation of a 50,000 record instance database execution time for these modules is typically on the order of hours. For example, the Reader may execute in one hour, the Restructurer in three hours, and the Writer in one hour.

There are a number of factors other than database size which affect the execution time of the Reader, Restructurer, and Writer. Execution time is obviously influenced by the complexity of the database(s) being translated and of the translation itself. This is reflected in the complexity and compatibility of the access paths to records (see Section 5.2). The percent utilization of space in the ADBMS databases used internally to the Data Translator also affects execution time. Time required for input/output operations on these databases rises dramatically when space utilization exceeds 90%. Hence, internal databases should be made at least twice as large as the minimum size to allow for at most 50% utilization at all times.

The incremental execution feature of the Data Translator (see Section 2.2.2.2) makes it possible for the user to break down the translation process into a number of increments, thereby controlling its execution. The Translator is a very flexible tool within its scope of translatable database types (sequential, ISP, and IDS).

2.3 User Interface

2.3.1 User Specifications

The user must provide, as input to the Data Translator, descriptions of the source and target databases and the required transformation between them. The high level languages used to specify this information are the augmented IDS MD sections (for describing the source and target databases) and the Translation Definition Language (for specifying the transformation).

2.3.1.1 The Extended IDS MD Section

Specifications of the source and target databases are input to the translator in the form of extended IDS MD sections. If a source or target database is an IDS database, the MD section will already exist. For ISP or sequential files, the MD section must be written according to specific rules. In either case, the IDS MD section must be extended to include certain information which is needed by the Data Translator, but which is unobtainable from a "straight" IDS MD section. Extensions to the MD section take the form of statements in a new level, the 61 level. The general form of an extended MD section is given in Figure 2-2. Level 61 entries are coded beneath the 02-49 field or group entry to which they apply.

```
01  record entry
    02  field entry
    02  field entry
    02  group entry
        03  group entry
        61  extension
            04  field entry
            04  field entry
        61  extension
    02  field entry
    61  extension
    61  extension
    61  extension
    61  extension
98  chain entry
98  chain entry

01  record entry
    02  field entry
    61  extension
    02  field entry
          .
          .
          .
```

Figure 2-2
A Generic Extended IDS MD Section

Extensions to the IDS MD section provide the following information:

a) Primary key definition. Each record type must have certain items designated as its primary key. The primary key for a record type consists of those items whose values, when taken together, uniquely identify each instance of that record type. Since it may not be possible to uniquely identify a record based on its own items alone, primary keys may include items from the record's owners (which the Data Translator uses as set significant items, see Section 4.1.2).

b) Identification of contained-in-repeating groups. A contained-in-repeating group is a repeating group within the physical confines of a record.

c) Phantom pointer relation identification. A phantom pointer relation is a technique of implementing an IDS chain without the control of IDS. User selected fields contain reference codes which point to other records. No 98 level chain is used to define the relation.

d) Match-key relation identification. A Match-key relation is another technique for implementing relations without IDS control. Two records are related when certain of their items' values match.

e) Specification of ISP/sequential record type identifier fields. Each ISP and sequential database record must contain an item whose value is unchanged in all instances of that record type. This item must be declared as the record type identifier field.

f) Identification of certain data types. Items whose usage is DISPLAY-1 or DISPLAY-2 must be identified.

2.3.1.2 The Translation Definition Language

The transformation between the source and target databases is specified by user-written Translation Definition Language (TDL) statements. The TDL is a block structured language [Figure 2-3].

```
TARGET RECORD
   TDLAP
      SOURCE RECORD
         ITEM
          .
          .
          .
      SOURCE RECORD
          .
          .
          .
   TDLAP
      .
      .
      .
TARGET RECORD
  .
  .
  .
EOF
```

Figure 2-3
TDL Block Structure

There is exactly one TARGET RECORD statement for each target record type. The TDLAP statements which make up a TARGET RECORD statement describe the access paths used to represent the target record in the source database. (An access path is a sub-structure of the source database which represents a target record type.) Each SOURCE RECORD statement describes a source record type on an access path. The purpose of the ITEM statement is to assign source items to target items and/or to specify an item value comparison which must succeed before the target record containing that item will be created.

It is not necessary to write a TDL specification for direct Reader-to-Writer translations.

2.3.2 Front End

Execution of the Data Translator requires a substantial number of control card files. Generating these files by hand can be a tedious, time consuming, and error prone process. To alleviate this problem, the Front End of the Data Translator will automatically build control card files and some other necessary files for the user. The Front End is run interactively and will prompt the user for information. This module will increase the user friendliness of the Data Translator.

2.4 Environment

2.4.1 Equipment and System Software Support

The Data Translator can be used on Honeywell Series 6000, 600, and 60 (Level 66) Information Processing Systems under the General Comprehensive Operating Supervisor. The following conditions must be satisfied to run the Data Translator.

1. IDS must be in use. ISP must be available to translate ISP databases.
2. IDS Data Query must be available.
3. One processor and at least 256 K of core must be available.
4. Secondary storage (disk) at least three times larger than the size of the source or target databases should be available.
5. One 9-track tape drive should be available.

In addition, translation should be performed as much as possible on a dedicated system.

2.4.2 Security

The Data Translator itself is unclassified. The Front End will prompt the user for the classification code to be used for all files which the Front End creates. It is the user's responsibility to insure the security of the databases used in a translation.

3.0 OVERALL DESIGN

Section 3 provides a general description of the overall structure of the Data Translator and the flow of data among its components. The components of the Translator are then grouped into three categories and described in more detail in Sections 4, 5, and 6. The language analyzers (IDS Analyzer and TDL Analyzer) are described in Section 4. The Translator execution modules (Reader, Restructurer, and Writer) are described in Section 5. Section 6 describes the Translator support modules (the Front End, ADBMS, and the ASP).

3.1 General Flow

The architecture of the Data Translator is illustrated in Figure 3-1 and the relationships among the various components are explained in this section.

The user written specifications of the source and target databases (Augmented IDS MD sections) are used separately as input to two independent runs of the IDS Analyzer which produce the source and target SDDL Tables. These and all other tables used by the Translator are stored as ADBMS databases. If multiple source databases are involved in the translation, their individual augmented IDS MD sections must be combined into one file. The same is true for multiple target databases. Hence, the IDS Analyzer is run once for the source databases and once for the target databases. The SDDL tables produced are used by Translator modules whenever information on the source or target databases is needed.

The user written source to target transformation (a Translation Definition Language specification) serves as input to the TDL Analyzer, which produces the TDL tables. The TDL tables are an internal representation of the TDL specification in a form processable by the Restructurer. Since both the source and target SDDL tables are used by the TDL Analyzer, it cannot be run until the SDDL tables are generated by the IDS Analyzer. In the case of a direct Reader-to-Writer translation the TDL specification and corresponding TDL tables are not needed.

Once the SDDL and TDL tables exist, the execution of the three main Translator modules can begin. The Reader is executed first and will convert the source databases into a logically equivalent ADBMS database

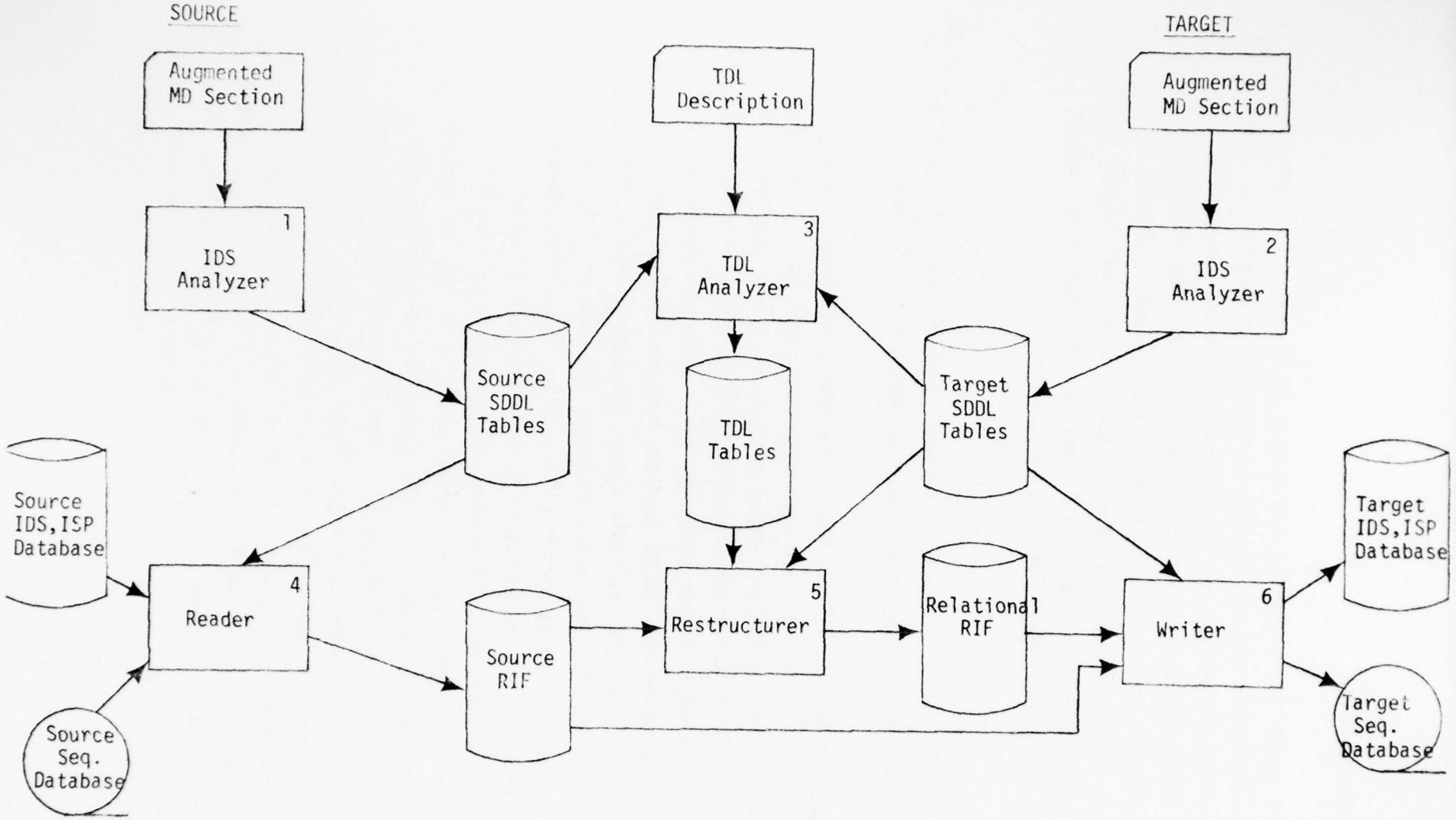


Figure 3-1
Version IIB Release 1 Architecture

called the Source Restructurer Internal Form (Source RIF). This is done so that the Restructurer always deals with a common database format.

If the Restructurer is executed it will create the Relational RIF. If the Restructurer is not executed (as in a direct Reader-to-Writer translation) there will be no Relational RIF. The three modes of restructuring (as discussed in Section 2.2.2.1) are depicted in Figure 3-2. In both complete and partial restructuring the Restructurer is executed. Although in the complete restructuring mode all record types are processed, in the partial restructuring mode only those parts of the Source RIF needed for restructuring are processed.

The form of the physical input to the Writer, the Target RIF, depends on the mode of restructuring employed. The Target RIF consists of either the Source RIF (for a direct Reader-to-Writer translation), the Relational RIF (for complete restructuring), or a combination of the two (for partial restructuring). The Writer adopts a consistent view of the Target RIF through the use of the Aggregate Schema Processor (ASP). The ASP permits the Writer to view an aggregation of multiple ADBMS databases (the Source RIF and Relational RIF) as a single database (the Target RIF)[Figure 3-3].

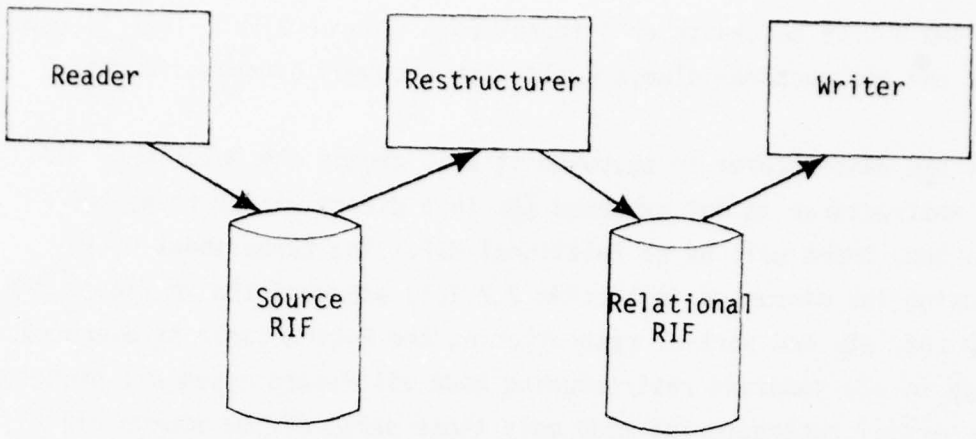
3.2 Database/Data Flow

This section summarizes the databases which are used internally to the Data Translator. Source and target IDS, ISP, or Sequential databases are excluded since they are primarily external to the Data Translator. Figure 3-1 clarifies the flow of data among the Translator modules. All tables are stored and processed by ADBMS.

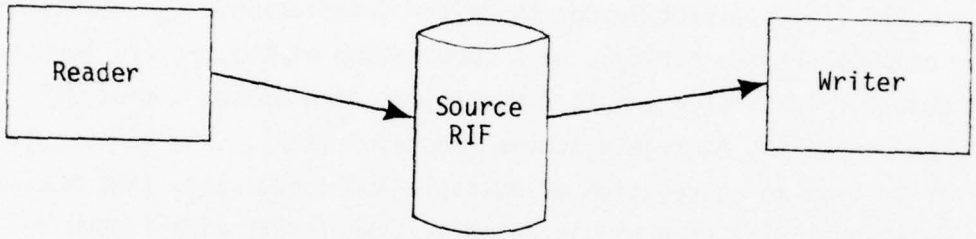
Source SDDL Tables

output by the IDS Analyzer
input to the Reader
input to the TDL Analyzer

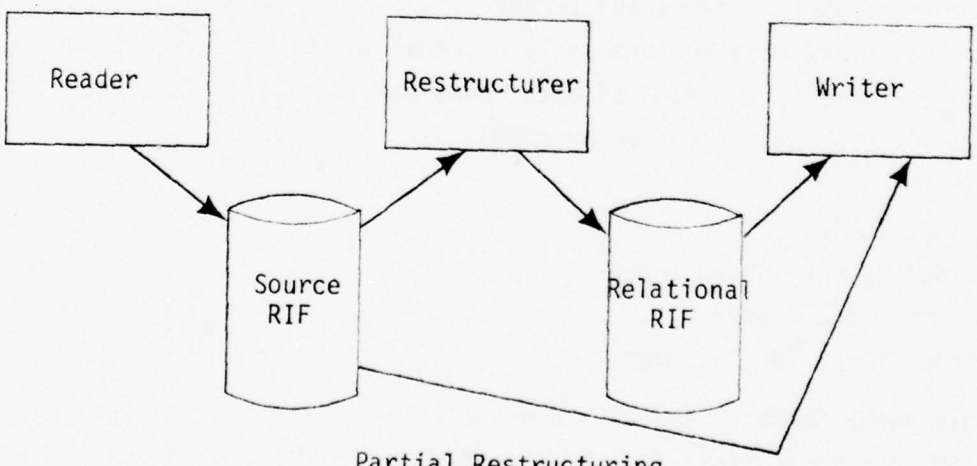
This ADBMS database contains a description of the source database(s) (IDS, ISP, and Sequential) for the translation. It is created by the IDS Analyzer from the user-supplied extended IDS MD section for the source database(s). It is used by the Reader, which requires a description of the source database(s) in order to access them. It is also used by the TDL



Complete Restructuring



No Restructuring



Partial Restructuring

Figure 3-2
Translator Modes of Operation

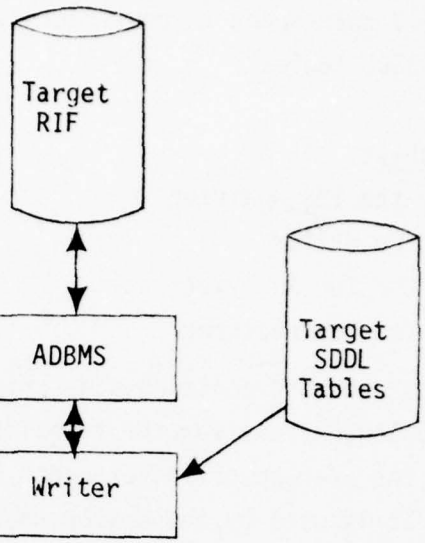


Figure 3-3a
Writer's View

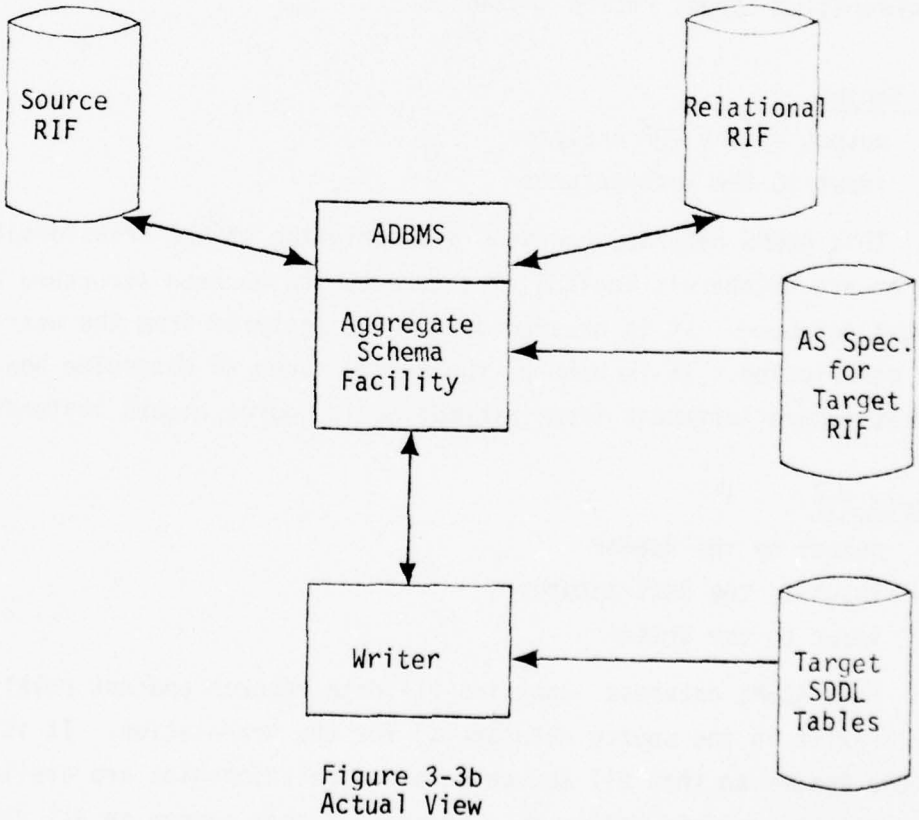


Figure 3-3b
Actual View

Figure 3-3
Input to Writer

Analyzer to verify that constructs specified in the user's TDL description are valid, i.e., correspond to constructs in the source database(s), as described the SDDL tables.

Target SDDL Tables

output by the IDS Analyzer
input to the Writer
input to the TDL Analyzer
input to the Restructurer

This ADBMS database contains a description of the target database(s) (IDS, ISP, and Sequential) for the translation. It is created by the IDS Analyzer from the user-specified extended IDS MD section for the target database(s). It is used by the Writer which requires a description of the target database(s) in order to generate them. It is also used by the TDL Analyzer to verify that constructs specified in the user's TDL description are valid, i.e., correspond to constructs in the target database(s), as described by the SDDL tables. Finally it is used by the Restructurer in generating target record instances.

TDL Tables

output by the TDL Analyzer
input to the Restructurer

This ADBMS database contains a description of the transformation from the source database's logical structure to the desired structure of the target database. It is created by the TDL Analyzer from the user-specified TDL description. It is used by the Restructurer to determine how to form target record instances from information in source record instances.

Source RIF

output by the Reader
input to the Restructurer
input to the Writer

This ADBMS database contains all data records and set relationships which exist in the source database(s) for the translation. It is created by the Reader so that all source data and relationships are available to the Restructurer and Writer in a standard format common to all translations.

The Restructurer accesses data from the Source RIF in order to create target data in the Relational RIF. In a partial restructuring translation, the Writer accesses the Source RIF in order to obtain data which is not restructured and hence does not exist in the Relational RIF.

Relational RIF

output by the Restructurer
input to the Writer

The Relational RIF is created by the Restructurer. In a complete restructuring translation, this ADBMS database contains all target data and relationships which must exist in the target database(s). In a partial restructuring translation, this ADBMS database contains only the target data and relationships which did not exist in the source RIF, i.e., all data and relationships which were generated by restructuring source data and relationships necessary for generation of the target database(s).

Target RIF

The Target RIF does not exist as a physical database, but rather, is a virtual database thought of as the composite of the Relational RIF and the portion of the Source RIF necessary to the Writer. It allows the Writer to operate on a standard view of the target data, regardless of the mode of restructuring which was used in the translation (see Section 6.3).

4.0 LANGUAGE ANALYZERS

The Data Translator uses two Language Analyzers - one to process the user's description of the source and target databases and the other to process the specification of the source to target database transformations.

4.1 IDS Analyzer Design

The first module in the specification phase of the translation process is the IDS Analyzer. It will take the user's description of the source or target database and convert it to the Stored Data Definition Language (SDDL) tables. The SDDL tables are subsequently used by all other Translator modules.

4.1.1 Purpose

The role of the IDS Analyzer is to compile/analyze the user's source or target database description (the augmented IDS MD Section) and to produce the internal SDDL tables used to drive the translation process. The IDS Analyzer must be executed twice, to produce two SDDL tables databases, one for the source database(s) and one for the target database(s).

4.1.2 Terminology and Concepts

The following is a list of most major terms used in describing the function and algorithm of the IDS Analyzer.

IDS Data Query	A Honeywell package used to convert the extended IDS MD Section into the IDS Data Query File, an IDS database which contains the extended MD Section as its data. The Data Query File is used as input to the IDS Analyzer.
Contained-in-repeating-group (CIRG)	A repeating group within the physical confines of the record.
Phantom pointer relations	A technique of implementing an IDS chain without IDS knowing about it. The user selects a field and places reference codes in it which point to other records. No 98 level chain is used to define the relation.

Match key relations	Another technique for implementing relations without IDS control. Two records are related together by having item values match.
Primary keys	Each record instance must be distinguishable from all other instances of its type. This is accomplished by extending the IDS MD Section to designate as primary key items those items whose values, when taken together, will uniquely identify all records of a record type. Since it may not be possible to uniquely identify a record based on its own items alone, primary keys may include items from the record's owners (which the Data Translator defines as set-significant items).
Set-significant items	Set-significant items are created by the Data Translator in each member instance to correspond to the primary key items in all owner instances related to that member. They are required to implement the Restructurer.

4.1.3 Input/Output

Figure 4-1 summarizes the major inputs and outputs of the IDS Analyzer process. Three steps will be involved in producing the SDDL tables: initialization of the IDS Query Dictionary database, population of the Query Dictionary, and execution of the IDS Analyzer. The function of each component in Figure 4-1 is summarized below.

Combined Extended IDS MD Section	Since the SDDL tables can describe up to five user databases, an extended MD Section must be prepared for each database. These are merged together by the user (resolving duplicate names and record id's) into one input file, the combined extended IDS MD Section.
IDS Query Dictionary	A machine-encoded, structured representation of the contents of the combined extended IDS MD Section suitable for input to the IDS Analyzer (even though this is not its usual use).
IDS QUTI	Utility program to initialize IDS databases.

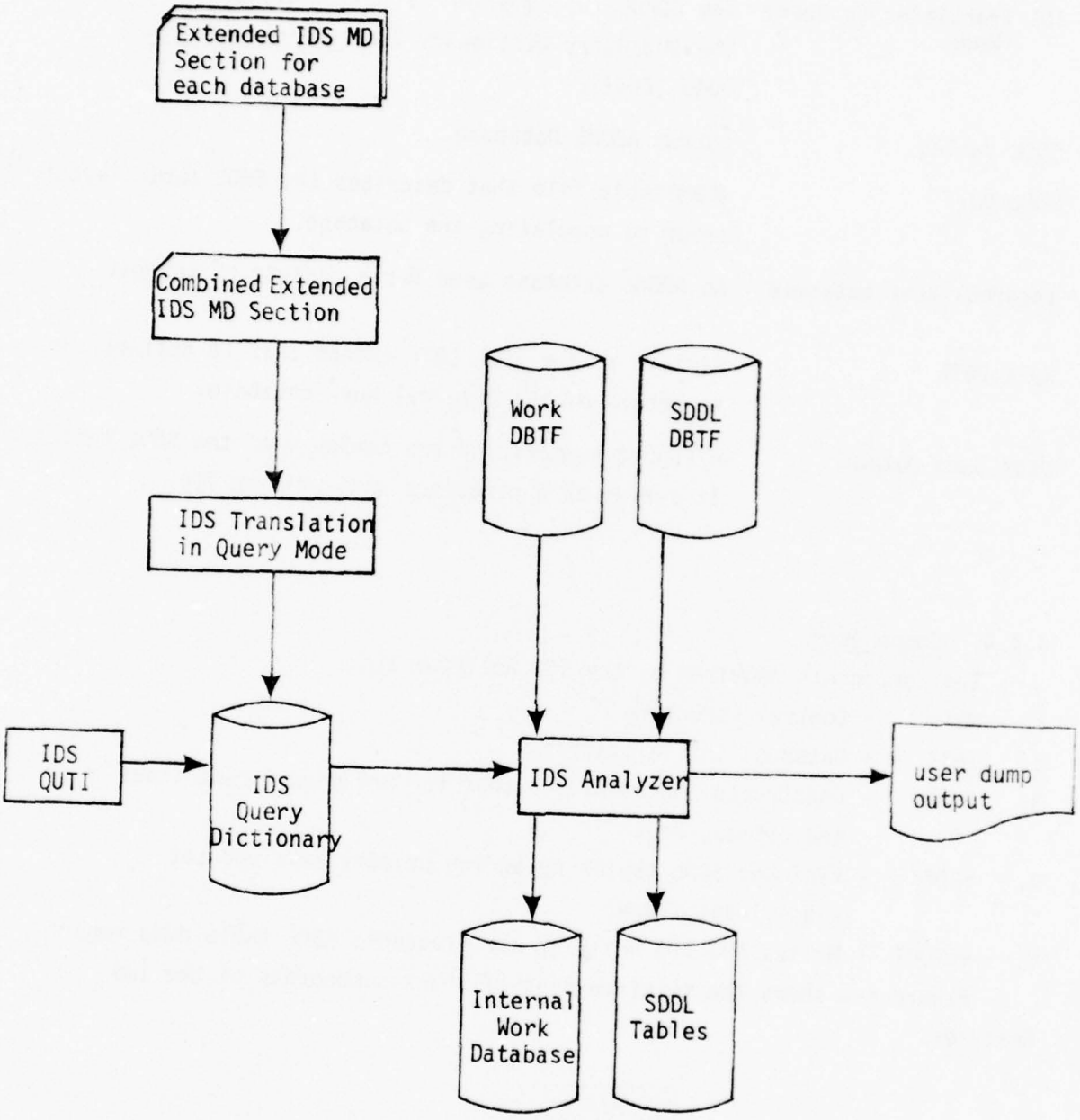

Extended IDS MD Section for each database
Combined Extended IDS MD Section
IDS Translation in Query Mode
Work DBTF
SDDL DBTF
IDS QUTI
IDS Query Dictionary
IDS Analyzer
user dump output
Internal Work Database
SDDL Tables

Figure 4-1

IDS Translator in Query Mode	The COBOL-IDS compiler ($ IDS) modified to build the IDS Query Dictionary (see IDS Data Query, DD46, DD47).
SDDL tables	Output ADBMS Database.
SDDL DBTF	ADBMStable file that describes the SDDL table layout prior to populating the database.
Internal Work Database	An ADBMS database used for a variety of things.
Work DBTF	Similar to the SDDL DBTF except that it defines the structure of the Internal Work database.
User Dump Output	A report summarizing the contents of the SDDL tables. It serves as a reference when writing TDL.

4.1.4 Components

The components required by the IDS Analyzer are:

1. Main - Control structure
2. INIT - Database initialization
3. IDSAN - Constructs SDDL tables except for set-significant items and primary keys
4. BLDPK - Finishes SDDL tables by adding primary keys and set-significant items
5. REPORT - Writes the IDS Analyzer user-readable SDDL table dump report

Figure 4-2 shows the relationships of these components of the IDS Analyzer.

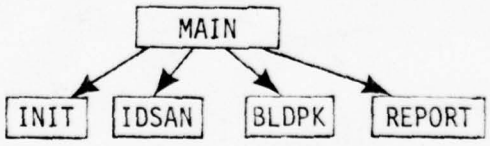


Figure 4-2 IDS Analyzer Main Components

4.1.4.1 MAIN Design

This component is very small, serving only to coordinate the other three components.

4.1.4.2 INIT Design

The two ADBMS databases, SDDL tables and Internal Work database are initialized using the contents of their respective ADBMS table files.

4.1.4.3 IDSAN Design

The IDSAN traverses the IDS Query Dictionary, building the appropriate SDDL table constructs.

4.1.4.4 BLDPK Design

BLDPK handles primary key items and set-significant items to complete the generation of the SDDL tables.

4.1.4.5 REPORT Design

A report summarizing the contents of the SDDL tables is produced.

4.2 TDL Analyzer Design

Completing the specification phase of a data translation is the TDL Analyzer process. It takes the user's description of the source to target transformations (expressed in the TDL Language) and transforms these specifications into a translator processable form - the TDL tables. The tables are used exclusively by the Restructurer module of the Translator.

4.2.1 Purpose

The role of the TDL Analyzer is to compile/analyze the user-supplied TDL description and produce the TDL tables. Using the SDDL tables, the TDL Analyzer performs error analysis and consistency checking of the TDL description.

4.2.2 Terminology and Concepts

The terminology used in the TDL Analyzer section is briefly described below.

Grammar	The structure and rules that describe the syntax of the language.
Reserved Word	A word in the language that has specific semantics associated with it, and hence is unsuitable for use as a user word.
Token	A basic symbol in the language such as word or punctuation.

The legal syntax of sentences in the TDL is defined by a grammar written in Backus-Naur Form (BNF). This BNF grammar will be processed by a grammar analysis program which produces a set of parsing tables. It is this set of tables which will drive the TDL Analyzer as it analyzes a TDL description. All of the legal sentences of the language will be encoded in these parsing tables.

4.2.3 Input/Output

The input/output relationships for the TDL Analyzer are depicted Figure 4-3.

The inputs to the TDL Analyzer will consist of the user-written TDL description, the parsing tables, source and target SDDL tables, and the TDL tables (an ADBMS database). The parsing tables are a set of tables derived from the TDL grammar rules which define the legal sentence forms. The source and target SDDL tables are output from the IDS Analyzer. The TDL tables database tables file is used to initialize the TDL tables database file which is produced during the TDL Analyzer run.

The output from the TDL Analyzer will consist of the TDL tables databases and a listing of the TDL description. The TDL tables are then input to the Translator during the execution phase. The listing of the TDL description will have error/warning messages where appropriate and a set of timing statistics at the end.

4.2.4 Components

Functionally, the TDL Analyzer consists of three main components, the Control, Syntactic, and Semantic modules. The Control Component performs initialization and wrapup. The Syntactic Component is responsible for analyzing the syntax of the TDL description, and constructing the TDL tables. The relationship of these three components is depicted in Figure 4-4.

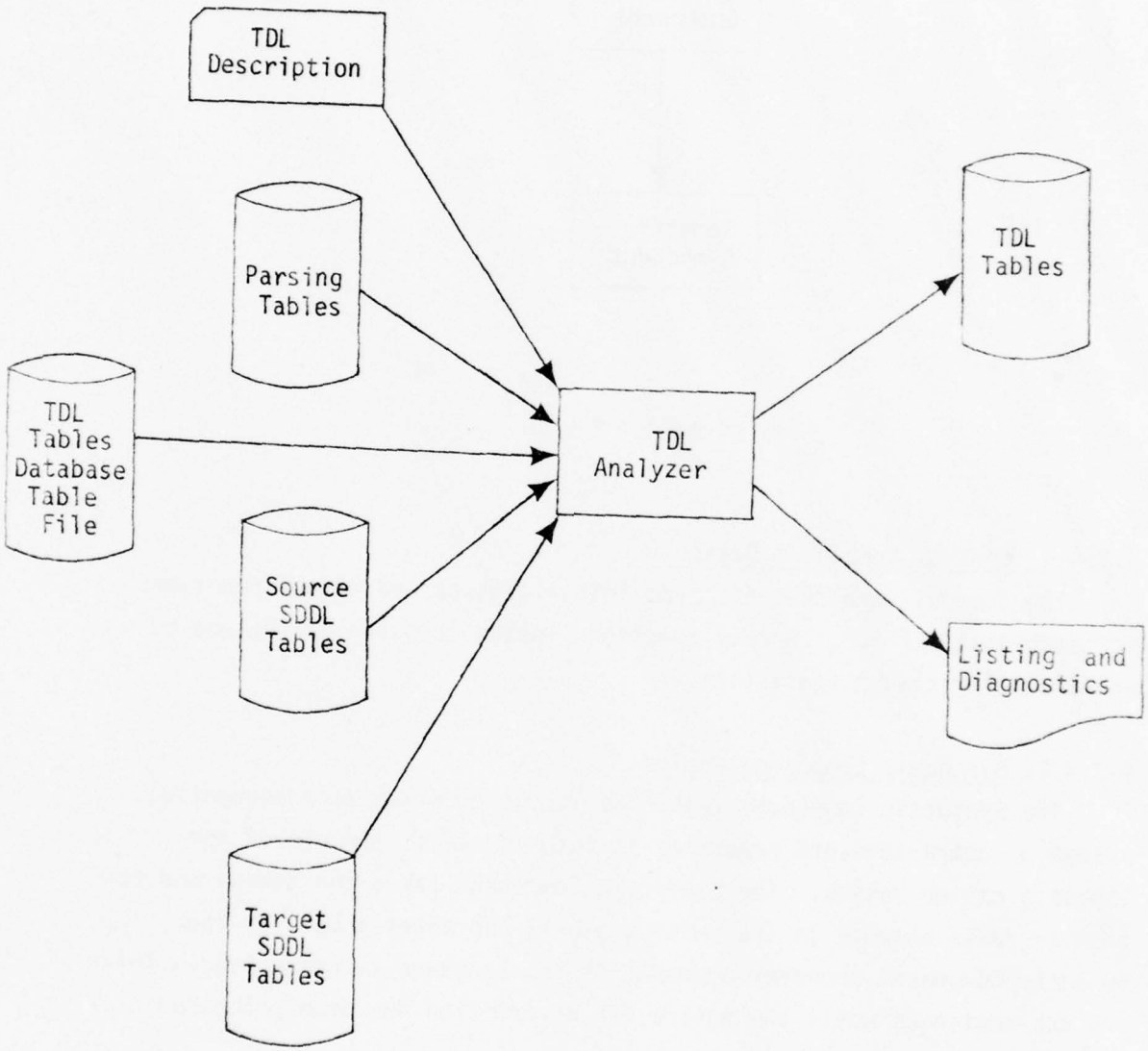

TDL
Description
Parsing
Tables
TDL
Tables
Database
Table
File
TDL
Analyzer
TDL
Tables
Source
SDDL
Tables
Listing and
Diagnostics
Target
SDDL
Tables

Figure 4-3

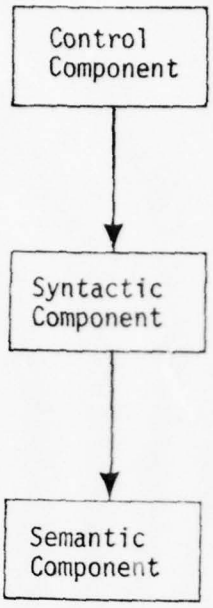


Figure 4-4

4.2.4.1 Control Component Design

The Control Component performs initialization and wrapup functions for each Analyzer run. Wrapup functions include closing all databases and providing timing statistics.

4.2.4.2. Syntactic Component Design

The Syntactic Component reads the TDL description as a sequential stream of characters and separates it into the basic symbols of the language called tokens. The Syntactic Component takes the tokens and the grammar rules encoded in the parsing tables and makes a call to the Semantic Component whenever a phrase of the language is recognized. This process continues until the entire TDL description has been processed.

4.2.4.3 Semantic Component Design

The Semantic Component builds the TDL tables from the information collected by the Semantic Components.

5.0 TRANSLATOR EXECUTION MODULES

After the successful completion of the specification phase of translation, the execution phase begins. Three modules comprise the execution phase - the Reader, Restructurer, and Writer.

5.1 Reader Design

The Reader is the first module of the Data Translator to access the user's database. It reads any combination up to five IDS, ISP, or WWDMS sequential databases. It can be run incrementally and its progress may be monitored through a job status file. The Reader is driven by information stored in the source SDDL tables.

5.1.1 Purpose

The Reader produces the source Restructurer Internal Form (RIF) database from the user's source database(s). The source RIF database is logically equivalent to the sum of the data in all source databases.

5.1.2 Terminology and Concepts

Contained-in-Repeating Group (CIRG)	A named collection of items in a record. In COBOL, a CIRG is defined as any field with an OCCURS clause.
Phantom Pointer Relation	A set or relation in an IDS database which is maintained by the user. Set membership is based on IDS reference codes stored in user data fields.

5.1.3 Input/Output

This section describes the inputs and outputs to the Reader.

Inputs:

Source Database	Sequential databases may be on tape, ISP databases require the ISP index file, and IDS databases must include all subfiles.
Source SDDL Tables	The description of the source database is stored in the source SDDL tables. It is an ADBMS database produced by the IDS Analyzer.

Accessor Object — Each Accessor (IDS, ISP, or Sequential) is compiled before every Reader run. This is done so that the IDS structure tables for the database being read will be available for the IDS Accessor. The ISP and Sequential Accessors are also compiled although they never change.

Outputs:

Report — The Reader produces a report which contains initialization information and a selected record dump of the SRIF. All error messages are also on this report.

Source RIF — The primary output of the Reader is the source RIF. It is the logical representation of the source database(s) which is processable by the Restructurer module.

5.1.4 Components

The Reader has seven modules.

1. Main Program - Performs major Reader initialization and controls the other components.
2. SRIF DDL Writer - The SRIF DDL Writer produces the ADBMS DDL statements for the SRIF using information stored in the source SDDL tables.
3. Table Initializer - The Reader's in-core tables are initialized by this module. The internal tables allow the Reader to access data stored in the SDDL tables in less time than going through the SDDL tables.
4. Data Movement - This module transfers the data from the source database to the source RIF.
5. Relation Linker - Links all the records together in the SRIF in the same manner in which they were linked in the source database.
6. Accessor - There are three Accessors, one each for sequential, ISP, and IDS databases. The Accessor retrieves every record from a database and returns it to the Data Movement or Relation Linker module.

7. Wrapup and SRIF Dump - The Wrapup module sets up the Reader for the next incremental run and prints the report.

The detailed relationships between the components of the Reader are very complex and are beyond the scope of this document.

5.1.4.1 Main Program Design

The Main Program sets up internal variables before calling either the Data Movement or Relation Linker modules.

5.1.4.2 SRIF DDL Writer Design

The first module to be executed is the SRIF DDL Writer. It produces the ADBMS DDL for all of the records, items, and sets in the source RIF. The records, items, and sets in the source RIF correspond to logically equivalent records, fields, and chains in the source database. The SRIF DDL Writer is executed only once (during the first Reader run). After the SRIF DDL Writer is finished, the SRIF file is initialized by the ADBMS Database Initializer.

5.1.4.3 Table Initializer Design

Processing time for the Reader is minimized by gathering the required information from the source SDDL tables once for all record and set types.

5.1.4.4 Data Movement Design

The Data Movement module calls the Accessor for a record from the source database. After determining the record's type, a record of the corresponding type is created in the SRIF and the data moved to it. Any necessary data conversion is performed before inserting the data in the SRIF. Any CIRGs in the source record are moved to a record in the SRIF. CIRGs which contain only null data are not moved to the SRIF.

5.1.4.5 Relation Linker Design

Once all records have been moved to the SRIF by the Data Movement module the Relation Linker creates in the SRIF all relations which were present in the source databases. For sequential and ISP databases, the algorithm is very simple because the records are retrieved in hierarchical order.

The IDS (network) relation linker is significantly more complex than the ISP/sequential linker. It connects the records in the SRIF together in the same order as they were along the IDS chain NEXT fields. Phantom pointer and match-key relations are also constructed by the Relation Linker.

5.1.4.6 Accessor Design

There are three Accessors; one each for sequential, ISP, and IDS databases. The sequential and ISP Accessors are FORTRAN programs which retrieve all records in a straight-forward manner. The IDS Accessor is an IDS-COBOL-GMAP program which is compiled before every Reader run with the MD section for the database being read. It uses information in the SICT and .QWRA to get every record in the database using the IDS RETRIEVE DIRECT verb.

5.1.4.7 Wrapup and SRIF Dump Design

This component of the Reader dumps the first and last records of every SYSTEM owned set so data values can be roughly validated. A Reader internal file is updated so succeeding Reader runs will know where to start in the database (if the Reader was not finished). The ADBMS set and record tables are written back to the source RIF to maintain currency across runs.

5.2 Restructurer Design

The Restructurer is the second of the three major modules of the Translator's execution phase. Assumming restructuring is to be performed, it must be executed after the Reader has completed and before the Writer can begin.

5.2.1 Purpose

The Restructurer is invoked to perform the logical transformations specified by the user's TDL description. The TDL description will specify how the source data is to be transformed into the desired target format. The Restructurer creates only target data records that are different from all source data records; any parts of the source database which will be in the same form in the target database as in the source database will not be transformed, but will remain in the source RIF database. Target data records created by the Restructurer will be stored in another standard

physical format, the relational RIF database. The Writer will then use the data from the source and relational RIF databases to write out the user's target database(s).

5.2.2 Terminology and Concepts

This section gives definitions of the terms and concepts used in the descriptions of the Restructurer algorithms.

Access Path	A hierarchical substructure of the source RIF database which defines a representation of a target record type in the source RIF. An access path must begin at the SYSTEM record of the source RIF schema.
Compatible	An access path is said to be "compatible below" a second access path if it shares a subtree, starting at the SYSTEM record, from which the primary key data values are drawn for the target record represented by the second access path. If this condition holds, it becomes advantageous to process both access paths simultaneously, since the same record instances are used by both of them.
Node	Each record on an access path is referred to as a node of the access path.
Stack	The Stack Builder module takes access paths from the TDL tables and sets up a data structure, similar to a stack, which is used by the Source Accessor to retrieve source record instances from the source RIF database. While somewhat more complex than a simple stack, this data structure will still be referred to as simply a stack throughout this section.
SYSTEM Record	Each ADBMS database has one instance of a record type known as SYSTEM placed in it by the Database Initializer. The SYSTEM record defines the "top" of the database schema; entry

into the data structure via ADBMS sets (relationships between record types) starts at the SYSTEM record. All access paths must also start at the SYSTEM record.

5.2.3 Input/Output

Figure 5-1 illustrates the inputs required and the outputs produced by the Restructurer. A brief explanation of each follows:

TDL Tables	An ADBMS database, output by the TDL Analyzer, in which are stored the translation specifications from the user's TDL description.
Source RIF Database	An ADBMS database, output by the Reader, which is logically equivalent to the user's original source database(s).
Restructurer Report	A report of the Restructurer execution.
Relational RIF Database	An ADBMS database in which the target record instances built by the Restructurer are stored. This database is then used by the Writer in producing the user's target database(s).
Target SDDL Tables	An ADBMS database, output by the IDS Analyzer, that describes the data structure of the user's target database(s).
Restructurer Work Database	An ADBMS database used by the DDL Writer to produce ADBMS DDL text.
Work Database Tables File	A sequential file containing the ADBMS DBTF for the Restructurer Work Database. This file is used to initialize the work database before it is used by the DDL Writer.
Relational RIF DDL Text	A sequential file, produced by the DDL Writer, containing ADBMS DDL statements for the relational RIF database. It also serves as input to the ADBMS DDL Analyzer.

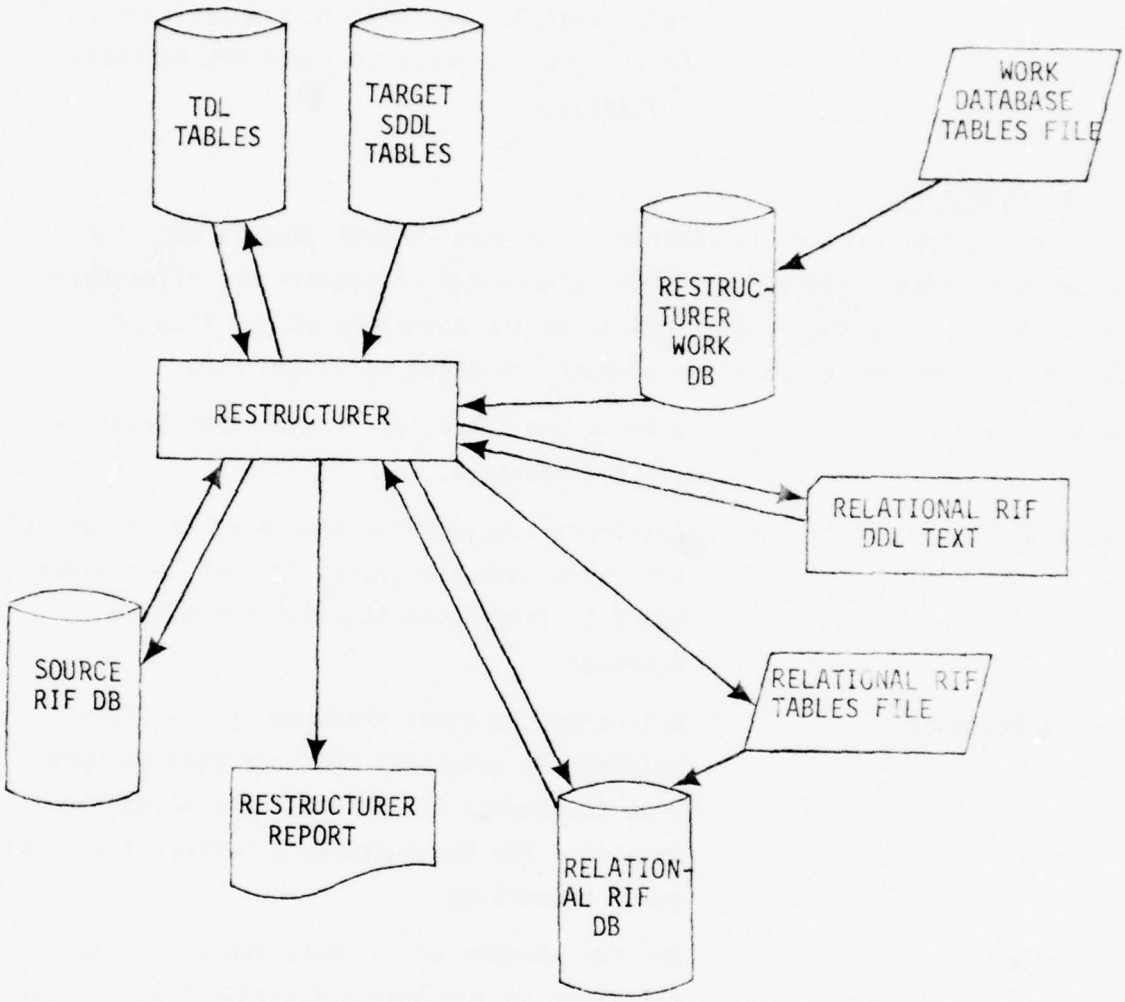


Figure 5-1
Restructurer Inputs and Outputs

Relational RIF Tables File	A sequential file, produced by the DDL Analyzer, containing the information from the relational RIF DDL text in tabular format. Also serves as input to the ADBMS Database Initializer.

5.2.4 Components

The Restructurer is implemented as a main control program and five major components. The relationshps between the components are illustrated in Figure 5-2 with the arrows indicating the direction of the flow of execution. A brief explanation of each component is given below.

Main Control	Directs the execution of the other Restructurer components.
Stack Builder	Retrieves access paths from the TDL tables and sets up a data structure, similar to a stack, which is later used to drive the Source Accessor.
Source Accessor	Driven by the stack produced by the Stack Builder, it retrieves the indicated records from the source RIF database and moves the data into the Restructurer's buffers for subsequent processing.
Qualifier	Decides whether or not each record instance retrieved by the Source Accessor fulfills the requirements specified in the user's TDL description.
Constructor	Retrieves the appropriate data values from the Restructurer buffers, constructs target record instances, and stores them in the relational RIF database.
Statistics and Wrapup	Writes out a summary of the Restructurer execution from data accumulated during the run.

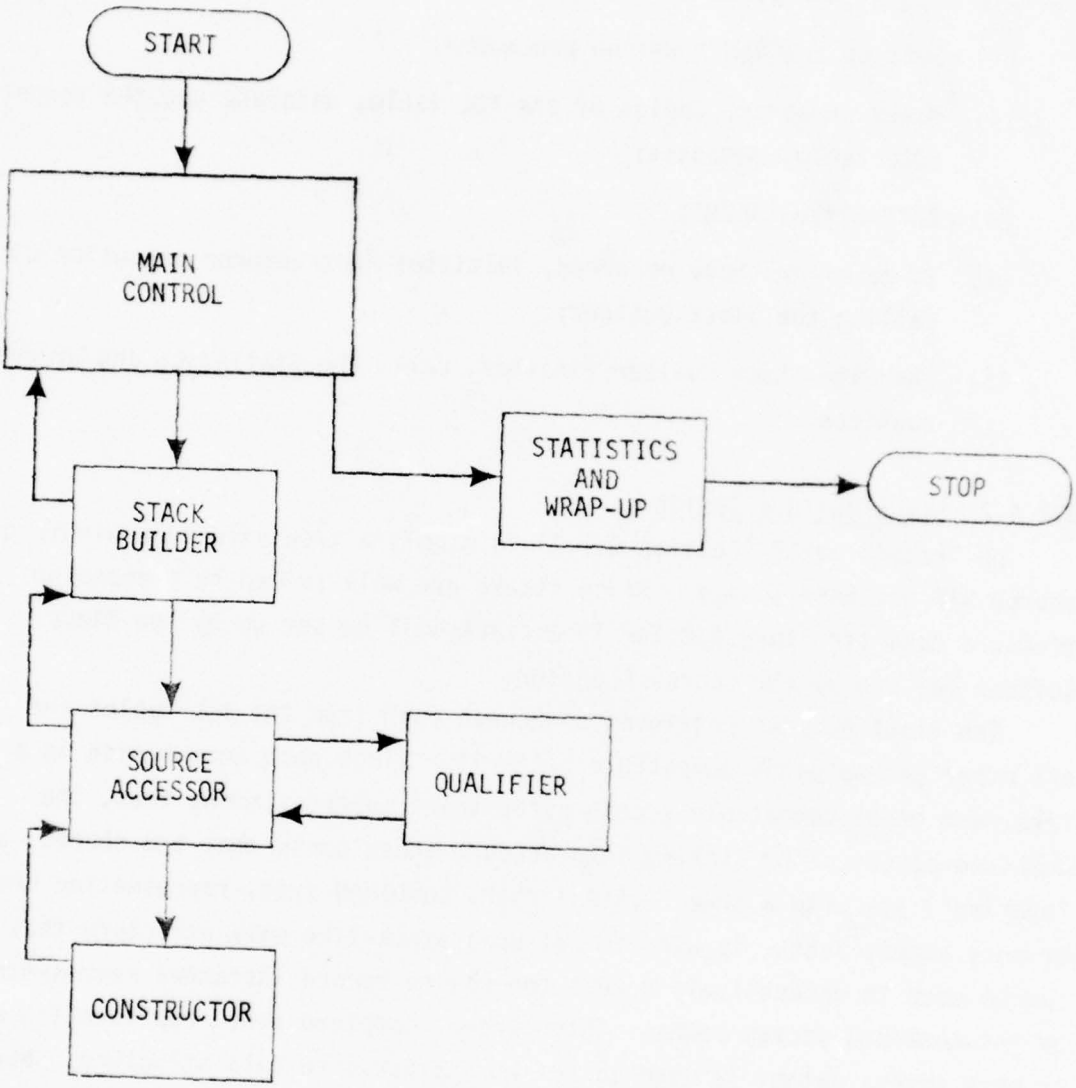


Figure 5-2
Major Components of the Restructurer

5.2.4.1 Main Control Design

The Main Control program oversees the Restructurer execution. It is a simple module that performs the following tasks sequentially:

(1) Sets up the abort wrapup procedure;

(2) Makes temporary copies of the TDL tables database and the target SDDL tables database;

(3) Initializes ADBMS;

(4) If no errors have occurred, initiates Restructurer execution by calling the Stack Builder;

(5) When the Stack Builder finishes, calls the Statistics and Wrapup routines.

5.2.4.2 Stack Builder Design

An "access path" (Section 5.1.2) is simply a tree structure within the source RIF database schema. Since stacks are well suited to processing trees, a data structure similar to a stack will be set up by the Stack Builder for use by the Source Accessor.

The Stack Builder retrieves an access path from the TDL tables and all other access paths compatible below it. Since each access path is a tree, and since compatible access paths share subtrees among them, the combined access paths (taking into account those parts that are shared, or "overlap") are also a tree. This larger, combined tree, representing one or more access paths, is used to set up a stack-like data structure that can be used to exhaustively access the source record instances represented by the combined access paths. This larger, combined tree, representing one or more access paths, is used to set up a stack-like data structure. When a stack has been set up, it is passed to the Source Accessor, which retrieves the source record instances represented by the combined access paths from the source RIF database.

When the Source Accessor finishes retrieving all record instances represented by a particular stack, the stack storage and internal buffer space are released, and the next stack is set up. If an access path is compatible below more than one other access path, it is only processed once; i.e., it may be skipped later when the second access path with which it is compatible is used to start a new stack. When all access paths in

the TDL tables have been used in a stack, the Stack Builder returns to the Main Control program.

5.2.4.3 Source Accessor Design

The Source Accessor uses the data structure set up by the Stack Builder to drive the record retrieval process. The source records needed to construct the corresponding target records are accessed from the source RIF database. As each source record instance is retrieved, it is passed to the Qualifier. The Qualifier consults the TDL tables and tests the source data to see if it satisfies all (if any) qualification criteria specified in the user's TDL description. If the record instance passes qualification, the Source Accessor continues by retrieving an instance of the next record type on the stack. If the record instance fails one or more qualifications, the next instance of the same record type on the stack is retrieved and the above process is repeated.

When a complete set of record instances for an access path on the stack has been retrieved, the Constructor is called. The appropriate source data values are moved from the Restructurer's buffers into a newly created target record instance; any required data type conversions are also performed and are not stored in the relational RIF database. When all target record instances that can be built using the current source record instances have been constructed, control is returned to the Source Accessor, which then retrieves the next set of source record instances so that new target record instances can be created.

When all source record instances for a stack have been exhausted, the Source Accessor returns to the Stack Builder.

5.2.4.4 Qualifier Design

Each record instance retrieved by the Source Accessor is passed to the Qualifier to determine if it satisfies all the qualifications specified in the user's TDL description. The Qualifier consults the TDL tables and tests all data item values in the source record instance that must be qualified for the access path currently being processed by the Source Accessor.

5.2.4.5 Constructor Design

The Constructor retrieves the necessary source data values from its buffers, performs any required conversions, and stores the data in the appropriate target items. The user may also provide a FORTRAN subroutine to perform unusual data conversion functions; any such routines are dynamically loaded and executed by the Constructor. When all target items have been assigned, the record is stored in the relational RIF database; duplicate record instances (i.e., instances whose primary key item values are the same as those of a previously stored record instance) are discarded.

5.2.4.6 Statistics and Wrapup Design

When all restructuring to be performed during the run has been completed, the Main Control program calls the Statistics and Wrapup routines to write out a report of the execution.

5.3 Writer Design

The final module of the translation process is the Writer. Its function is to produce the target database for subsequent normal use by the database administrator who initiated the whole translation process.

5.3.1 Purpose

The most important role of the Writer is to produce a target database according to the user's specifications. Two types of databases may be directly produced; IDS database and sequential file. The sequential file is subsequently sorted by SORT/MERGE to yield a standard WWDMS sequential database, or if ISP is desired, the utility XUTIL is loaded to generate an ISP database.

In producing a target database, two goals must be met by the Writer. These are:

1. Preserve the logical relationships present in the internal form databases produced by the Reader and Restructurer. The effects of all TDL-specified transformations between the SRIF and the RRIF must remain intact when the user's target database is produced.

2. For IDS databases, the target database must, of course, be a legal IDS file, but equally important, the Writer will not produce (or propagate) poor database design techniques; specifically,

contained-in-repeating groups, phantom pointers/chains and match-key relations will not be allowed in a target database.

Some of the more detailed features necessary in the Writer are listed below:

1. The Data Translator internal form databases can hold up to five user target databases which may be of differing types. Although the Writer can only output one database at a time, it should have the ability to select any one of the databases within the internal form (RIFs) which the user specifies.

2. A debug feature will be provided which will enable the user to obtain a snapshot look at every record instance written. This will facilitate the checking of the TDL without resorting to WWDMS or application programs.

5.3.2 Terminology and Concepts

The following is a formal list of most of the major terms used in reference to the Writer.

Writing	A verb used to indicate the process of transferring records from the Target RIF to the target (user) database.
Aggregate Schema Data Definition Language (ASDDL)	A special DDL, created by the DWTR (see below) which is a super-set of the DDLs that individually describe the Source RIF and Relational RIF. It serves as the link between the user target record, item and set names, and the internal form record, item and set names.
Aggregate Schema DDL Analyzer (ASDDLA)	A program that accepts input ASDDL statements and produces a set of tables used by the Aggregate Schema Processor (ASP, see below). The tables contain all the information needed by the Writer to correctly locate in the Source RIF or Relational RIF the record and set instances needed in the target database.

Aggregate Schema Processor (ASP)	A collection of routines which logically sit on top of ADBMS routines. The Writer uses ASP routines to retrieve record instances, item values, or to traverse sets. ASP, in turn, determines which ADBMS calls must be made (remember that the Source RIF and Relational RIF are ADBMS databases) in order to satisfy the requests.
DDL Writer, Target RIF (DWTR)	This component writes the ASDDL statements.

5.3.3 Input/Output

There are four modes to the Writer and one wrapup mode for ISP/sequential target databases. Each is defined below.

1. IDS target, Reader-Restructurer-Writer translation.
   This is the regular mode for target IDS databases; e.g., record or relation instances were changed substantially enough to warrant using TDL.

2. IDS target, Reader-Writer translation.
   Used for direct Reader-to-Writer translations, it is the same as #1 except that input files which would have been created by the Restructurer or TDL Analyzer do not exist in this configuration.

3. ISP/sequential target, Reader-Restructurer-Writer translation.
   Regardless of whether the user wishes an ISP or sequential target database, the Writer will output an unsorted sequential file. It is the user's responsibility to insure that record instances have the correct sort key values via the TDL (and probably user routines).

4. Sorting the sequential output (and for ISP only, loading an ISP file).
   For sequential output, the file must be sorted. The sorted file may then be used to load an ISP database. Although the programs used to perform these functions are Honeywell utilities, the Data Translator provides the control cards to run the utilities.

The major inputs and outputs to the Writer module are summarized below.

IDS Database	The target file(s) output by the Writer. It may consist of multiple areas, or multiple subfiles. Created by the user with the desired page ranges, page sizes, inventory, etc.
QUTI	The first activity of the writing process. For the initial incremental run only, the IDS database must be initialized by the QUTI activity. A separate activity is required for each area.
Relational RIF	Output ADBMS database from the Restructurer. It contains all "changed" record instances. This file is not present for direct Reader-to-Writer translation.
Source RIF	Output ADBMS database from the Reader. All source database(s) record instances are present here.
Target IDS MD section	Written by the user, it is the DDL for the target database. All records, items and chains are defined exactly as the user wishes them to be in the database. Physical design parameters chosen in the target MD section are basically irrevocable once the Writer has executed.
Source & Target SDDL Tables	Output ADBMS databases from the IDS Analyzer. The Source SDDL tables are used by the DWTR while the Target SDDL tables are used by almost all Writer components.
Restructurer Work Database	Output ADBMS database from the Restructurer. This file is used by the DWTR and is not present in Direct Reader-to-Writer runs.

TDL Tables	Output ADBMS database from the TDL Analyzer. Used as input to the DWTR. It is not present for Direct Reader-to-Writer runs.
Report	Writer execution report detailing the past history of previous incremental runs, a summary of the records to be written on the current run and the results of storage (time, number of instances) for each record type.
Errors	Writer error report.
Sequential Output	A linked file or tape used to write a sequential (unsorted) database. A separate physical file is required for each incremental run.

5.3.4 Components

The Writer components are executed sequentially as listed below with the main program handling all control.

1. Main Program - Controls all writer components and decides whether or not to continue when errors occur.
2. SETUP - Determines parameters for this Writer run.
3. DWTR - Writes the ASDDL.
4. ASDDLA - Analyzes the ASDDL to produce ASP tables.
5. INIT - Performs initializations necessary for the writing process.
6. PHASE1 - Writes out all records in the target database.
7. PHASE2 - Closes all files and writes a new copy of the between runs save file.

Figure 5-3 shows the relationships of the components of the Writer.

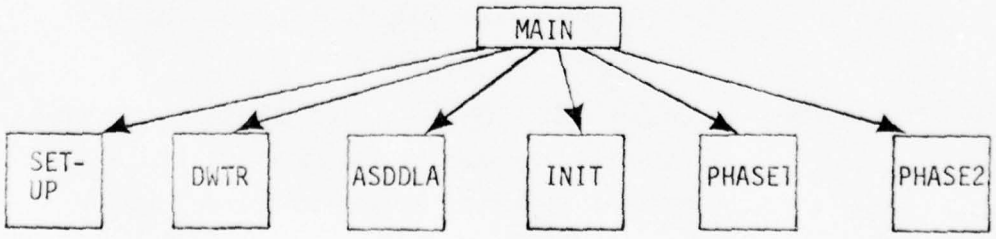


Figure 5-3

5.3.4.1 Main Program Design

The function of the main program is to provide a controlling structure to the entire writing process. After each routine is called and has returned, its return code is checked. The main program attempts to forge ahead with Writer execution despite errors in an attempt to discover as many mistakes as possible in one run. If this is not possible due to the severity of the mistake, the Writer is shut down.

5.3.4.2 SETUP Design

The SETUP module gathers necessary information such as what database is to be written and what mode of the Writer is to be used.

5.3.4.3 DWTR Design

As previously mentioned, the DWTR writes ASDDL which defines where target records, sets, and items are located (e.g., Source RIF or Relational RIF). All ASDDL statements are written to a temporary file for use by the ASDDLA.

5.3.4.4 ASDDLA Design

The Aggregate Schema DDL Analyzer (ASDDLA) is fully described in Section 6.3 of this manual. The Writer calls the ASDDLA to analyze the ASDDL produced by the DWTR. Upon completion, everything is ready for use by the ASP.

5.3.4.5 INIT Design

Final initialization functions are performed such as opening ADBMS databases, determining the record types involved in this Writer run, etc.

5.3.4.6 PHASE1 Design

All records in the target database are stored. Record instances are built one data item at a time because item conversion may be necessary. For the case of sequential file output, writing should present no difficulties; the records retrieved from the Target RIF are simply passed to the standard I/O routines. Since the records presumably have the correct sort key values, the order of record instances output is irrelevant as they must be subsequently sorted anyway. However, for IDS databases, it is a different problem. The

Writer uses IDS to perform the actual physical storage of record instances. Similarly, IDS does all the chain linking. Because IDS is doing the low level work, the Writer must insure that all IDS routines are "guided" into executing as desired.

5.3.4.7 PHASE2 Design

Final wrapup steps are performed, such as closing ADBMS databases and writing a new Between Runs Save File with the history of all incremental runs to date.

6.0 TRANSLATOR SUPPORT MODULES

6.1 Front End

6.1.1 Purpose

The Front End module is an interactive program intended to increase the user-friendliness of the Data Translator. This module provides a means to automatically build control card files for the IDS Analyzer, TDL Analyzer, Reader, Restructurer, and Writer. The user will have the option of building the control card files needed for an entire translation in one terminal session or in a series of terminal sessions.

6.1.2 Terminology and Concepts

An ADBMS database will be used by the Front End to keep the MPCCF (Modified Prototype Control Card Files) and other information needed to create files and provide data to the Control Card Drivers. The MPCCF will consist of both control cards and control statements. The control cards will form the skeleton on which the final control card file is based. The control statements, on the other hand, will be statements written in the Front End's internal macro language which are interpreted within the Front End to control the generation of the actual control card file. There are approximately twenty different MPCCFs in the database, one for each basic control card set up possible.

6.1.3 Input/Output

The Front End will receive input from two sources; the ADBMS database, and the user's responses at the terminal. The user will be prompted only for information that is actually needed for the given translation.

The output of the Front End will consist of one or more control card file(s). The control card files should be ready to run without modification. Diagnostics will also be produced.

6.1.4 Components

The Front End consists of four major logical components as shown in Figure 6-1; the Main Program, the Initialization routine, a group of Control Card Drivers, and a Control Card Generator.

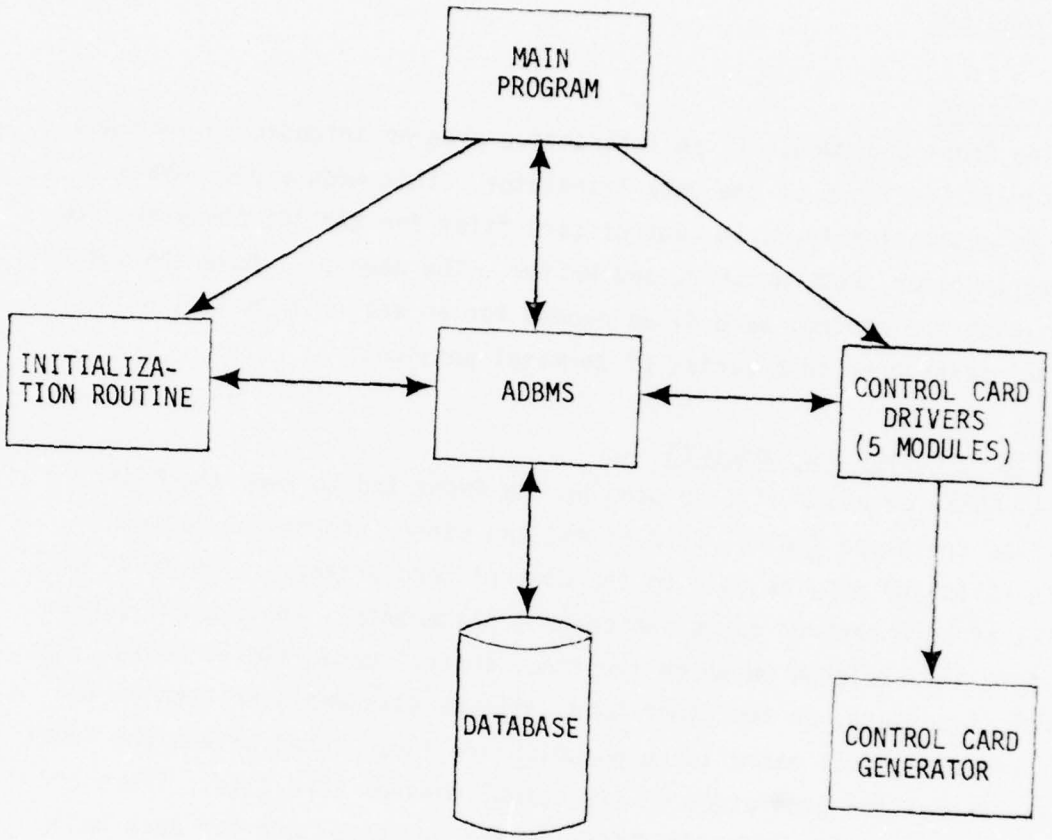


Figure 6-1
Overview of Front End Components

The Main Program will do some bookkeeping and will control the overall flow of the Front End. The Initialization routine will be called only once for each Translation. There will be a Control Card Driver component for each of the five Translator modules; IDS Analyzer, TDL Analyzer, Reader, Restructurer, and Writer. The components in this group will prompt the user for information particular to the respective module. Each Control Card Driver will call the last major component, the Control Card Generator, to actually generate the control card file. A more detailed description of each component is contained in subsequent sections.

6.1.4.1 Main Program Design

The Main Program serves three main functions. The first is to interact with ADBMS and determine if initialization has been previously accomplished, and if not, to call the Initialization routine. The second function is to determine the Translator module for which the user desires to build control card file(s). The last main function is to print on the terminal a list of the control card files that the Front End has built.

6.1.4.2 Initialization Design

The Initialization routine serves two main functions. The first is to request information from the user that will be used in all of the control card files. The second main function is to create files that will be needed for the translation and to print a list of these files. The Initialization routine is called only once for a translation since the information will be stored in the ADBMS database and used in subsequent terminal sessions.

6.1.4.3 Control Card Drivers Design

There will be five Control Card Drivers, one for each of the Translator modules, but the functions they perform will be basically the same. A Driver will put the information needed by the Control Card Generator into the form it can use, and then call the Generator to put the control cards into a temporary file. The Driver will then create a permanent file and copy the control cards into it.

As input, the Drivers will use both the ADBMS database and user responses to prompts. The final output of a Driver is one or more control card file(s).

6.1.4.4 Control Card Generator Design

The input to the Generator is provided by one of the Drivers. The call to the Generator also specifies which MPCCF is to be used. After the proper MPCCF is selected, card images are read and processed by the Generator. The process is driven by the input from the Driver, resulting in one of many possible variations of the same MPCCF. The result is a temporary file containing the desired control cards.

6.2 ADBMS

ADBMS is a database management system which facilitates the creation, maintenance and accessing of simple and complex data structures. It consists of a collection of FORTRAN-CALLable subroutines whose purpose is to create databases from a user's data structure description, and to serve as an interface between the user and these databases. The following components make up the environment in which ADBMS is used:

a) The database itself, containing the data which is to be accessed and a tabular representation of its schema.

b) The database control system, ADBMS.

c) The user's database access program, containing CALLs to ADBMS routines which access the database.

6.2.1 Purpose

ADBMS is used in the following modules of the Data Translator:

IDS Analyzer

a) creates SDDL tables (ADBMS database)

b) uses Internal Work Database (ADBMS database) internally

TDL Analyzer

a) retrieves information from source and target SDDL tables (ADBMS databases)

b) creates TDL tables (ADBMS database)

Reader

a) uses source SDDL tables (ADBMS database) as input

b) creates source RIF (ADBMS database)

c) uses DDL writer work database (ADBMS database)

Restructurer

a) retrieves data from source RIF (ADBMS database)
b) stores data in relational RIF (ADBMS database)
c) accesses TDL tables (ADBMS database)
d) uses DDL writer work database (ADBMS database)

Writer

a) uses target SDDL tables (ADBMS database) as input
b) uses relational RIF (ADBMS hash database) and source RIF as input
c) uses TDL tables (ADBMS)

In the environment of the Data Translator, the ADBMS "user" is the Data Translator and the Translator modules act as database access programs.

6.2.2 Terminology and Concepts

The terminology and concepts of ADBMS are described below.

Currency	ADBMS currency indicators are used as place markers to keep track of the state of the interface between the user program and the database.
Database	An initialized ADBMS database is a random file consisting of formatted physical pages on which all information in the database is stored. The database tables are stored on the first page(s) of the database. The remaining pages are initialized to a specific format.
Database Key	The unique identifier which distinguishes a record instance from all other record instances in the same database.
Database Tables (DBT)	Tabular form of the logical description of a database according to its DDL. The database tables exist physically in two forms: 1) on the first page(s) of the corresponding initialized database random file, and 2) as a separate sequential database tables file.

Database Tables File (DBTF)	The intermediate sequential form of the database tables, used within the DDLA/DBINT.
Database Description Language (DDL)	A language used to describe the schema of an ADBMS database in terms of records, items, and sets. A specific database description written in this language is also called a DDL.
DDL Analyzer/Database Initializer (DDLA/DBINT)	The utility module used to create ADBMS databases. It analyzes the DDL and if there are no syntax errors or inconsistencies, initializes an ADBMS database according to the DDL description.
Item	The elementary data unit in the database; used to represent specific data as a number, a string of characters, a logical truth value, etc.
Multiple Databases	ADBMS has the capability to manage multiple databases simultaneously, i.e., operations can be performed first on one database and then another without closing the first database and opening the second database between operations.
Ordered Set	The member-to-owner relationship for an ordered set is established when a member is "added" to the set. This operation is requested by the user but implemented and maintained by ADBMS. The sequence of retrieval of member records and the linkage of new member records in the set are controlled by the set ordering criterion (specified in the DDL).
Record	A named collection of data items used to represent the major entities of an application.
Schema	The description of the logical structure of a database.

Set	A named collection of record types which specifies an ordering or relation among the records within it.
SYSTEM Record	A predefined non-hash record type which is implicitly included in every database schema description.

6.2.3 Input/Output

The input/output relationships for running the DDLA/DBINT are given in Figure 6-2. The DDL consists of statements which express the schema of an ADBMS database in a form recognizable to the DDLA/DBINT. The hash input consists of statements which cause user-specified values to overide default values of hashing function parameters. The DDL and optional hash input are input to the DDLA/DBINT which produces an initialized database file and, optionally, the sequential database tables file.

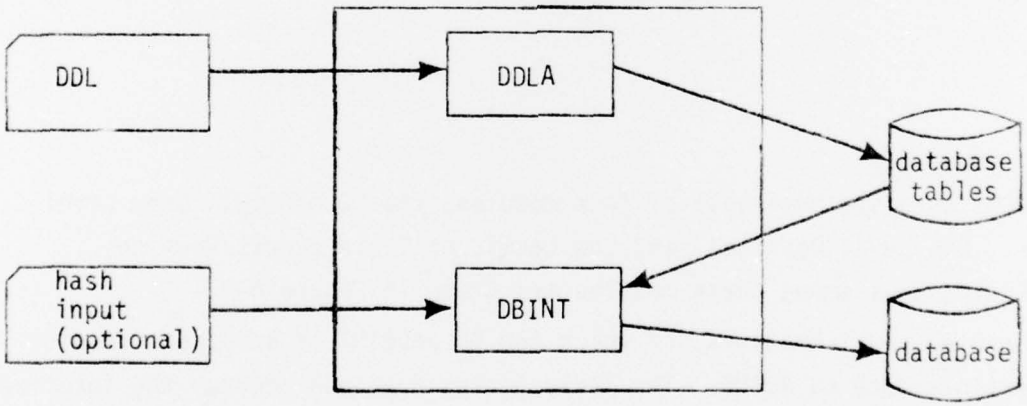


Figure 6-2
DDLA/DBINT

The ADBMS scenario for running a user's database access program is given in Figure 6-3. User database access programs contain calls to ADBMS routines which perform the actual database access functions requested.

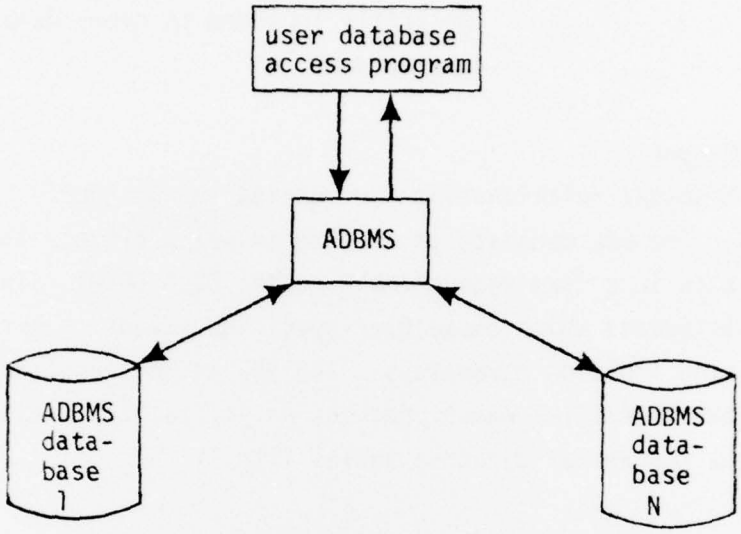


Figure 6-3
ADBMS Scenario

6.2.4 Components

ADBMS consists basically of four modules, the DDLA/DBINT, User Level Routines, Mid Level Routines, and Low Level, or Table Access Routines. The relationships among these modules are given in Figure 6-4.

The User Level Routines, of which the DDLA/DBINT is a subset, provide the user interface to ADBMS. The Table Access Routines provide the interface between ADBMS and the database tables. The intra-ADBMS relationships among the modules are somewhat looser, with access to the database being performed at all levels.

6.2.4.1 DDLA/DBINT Design

The DDLA stage reads the DDL, recognizes individual statements, and builds the appropriate control blocks in the database tables to store and maintain the information specified in the DDL. The output of the DDLA

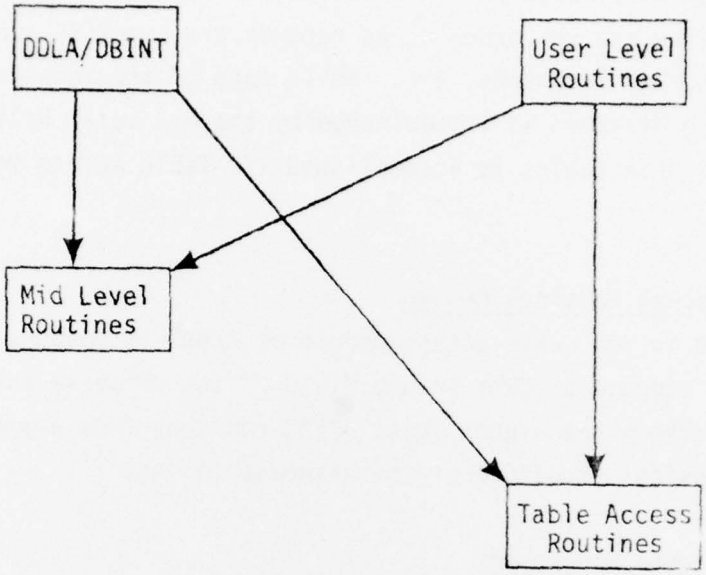


Figure 6-4

module is the DBTF. The DBTF and the optional hash input are the inputs to the DBINT stage of the DDLA/DBINT. The DBINT produces an initialized database ready for the storage and accessing of data.

6.2.4.2 User Level Routines Design

The interface between the user and the database in ADBMS is accomplished via CALLs from the user program to a set of ADBMS routines designated as being "user level". Each user level routine corresponds to an accessing function which the user may request to be performed on the database. Each routine translates this high level request into physical input/output operations on the database, using information stored in the database tables and the database itself. In general, the actual interface with the database and database tables is accomplished via the routines in the mid level and table access modules.

6.2.4.3 Mid Level Routines Design

Mid level routines in ADBMS perform such functions as database storage allocation, managing database pages in and out of core as they are accessed, maintaining currency and set ordering as records are accessed and members added to and retrieved from sets, etc. While much of the interface with the actual data in the database is accomplished by the Mid Level Routines, all access to the database tables is accomplished via Table Access module routines.

6.2.4.4 Table Access Routines Design

Each routine in the table access module of ADBMS is responsible for the retrieval or storage of data in one field of the database table control blocks. This isolates the higher level ADBMS routines from any design change in the physical structure of the database tables.

6.3 Aggregate Schema Processor

The ASP is an extension of ADBMS which, when used in conjunction with ADBMS, allows the user to view an aggregation of subsets of one or more ADBMS databases as one Aggregate Schema (AS) database and thus access multiple databases as one database.

The ASP environment, then, consists of the following elements:

a) Several ADBMS databases, each containing the data to be accessed and a tabular representation of its schema.

b) ADBMS.

c) The AS Database Tables, a tabular representation of the AS Database's logical description.

d) The Aggregate Schema Processor (ASP), consisting of a collection of FORTRAN CALLable subroutines whose purpose is to map AS database access requests, based on the information stored in the AS database tables.

e) The user's database access program, containing CALLs to ASP routines.

6.3.1 Purpose

The Data Translator uses the ASP in the Writer module to facilitate accessing the Source RIF and Relational RIF ADBMS databases. The Writer is essentialy the ASP user, or database access program in this case.

6.3.2 Terminology and Concepts

The terminology and concepts of the ASP are described below. Many ADBMS terms and concepts are also fundamental to the ASP, while other terms have different meanings in the two contexts; reference to Section 6.2.2 may be helpful.

Aggregate Schema (AS) Database	A logical "view" of one or more ADBMS databases. The term is used in a sense to refer to the composite of a) several ADBMS databases, and b) the ASDBT.
AS Database Key	The unique identifier which distinguishes an AS record instance from all other record instances in the same AS database.
AS Database Tables (ASDBT)	Tabular form of the logical description of an AS database according to its ASDDL and ASDNDDL.
AS Data Definition Language (ASDDL)	A language used to describe an AS database view in terms of sets, records, and items and their mappings to corresponding ADBMS database constructs. A specific AS database description written in this language is also called an ASDDL.
ASDDL Analyzer (ASDDLA)	A utility module used to create AS databases. It analyzes the ASDDL and if there are no syntax or semantic errors, produces the ASDBT according to the ASDDL description.
AS Database Name DDL (ASDNDDL)	A series of statements which identify the ADBMS databases involved in an AS database.
Assigned Record	A one-to-one correspondence between the AS record type and schema (ADBMS) record type.

	such that the schema record is viewed directly through the AS assigned record type.
Assigned Set	A one-to-one correspondence between the AS set type and a schema (ADBMS) ordered set type.
Coupled Record	A coupling of two schems (ADBMS) record types such that the AS coupled record type is viewed as a merge of the two schema record types.
Currency	The ASP uses ADBMS currency indicators in addition to its own currency indicators to keep track of the state of the interface between the user program and the database.
Item	AS item types defined in an AS record type specify which schema (ADBMS) items in the corresponding schema (ADBMS) records are to be included in the AS database view of the record and how they are to be viewed.
Match-key set	AS match-key set types are parallel to ADBMS match-key set types, but allow the owner and member record types to reside in different ADBMS databases.
Record	AS record types specify which schema (ADBMS) record types are to be included in the AS database view and how they are to be viewed.
Set	AS set types define which schema (ADBMS) sets are to be included in the AS database view and how they are to be viewed. They may also define new match-key sets which are not defined at the schema (ADBMS) level.
Set significant items	Similar to ADBMS set significant items.

6.3.3 Input/Output

The input/output relationships for running the DDLA/DBINT are given in Figure 6-5. The ASDDL consists of statements which describe the Aggregate Schema database view of its underlying ADBMS databases. The ASDNDDL consists of statements which identify each ADBMS database involved in the AS database. The ASDDLA inputs both the ASDDL and the ASDNDDL and references the database tables of each of the underlying ADBMS databases to produce the ASDBT as output.

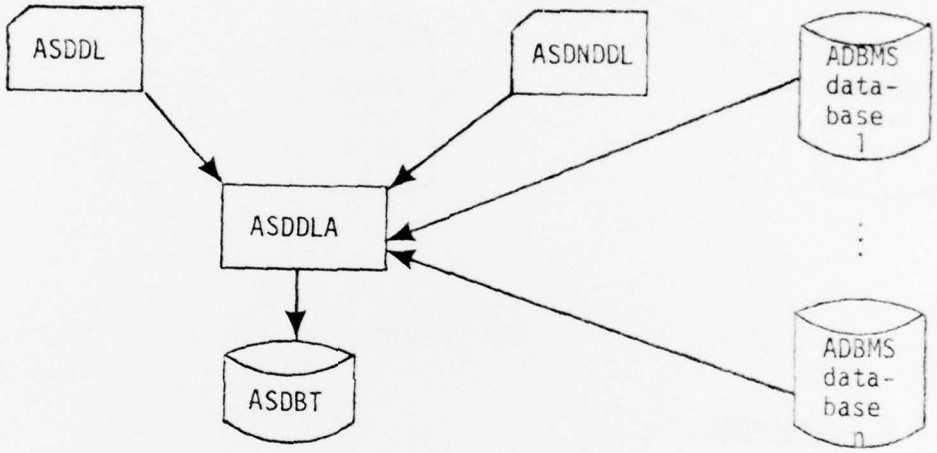


Figure 6-5
The ASDDLA

The scenario for running a user's database access program under the ASP is given in Figure 6-6. User database access programs contain CALLs to ASP routines. These routines use the information stored in the ASDBT, the ASDNDDL, and the individual ADBMS databases themselves to perform the actual database access functions requested.

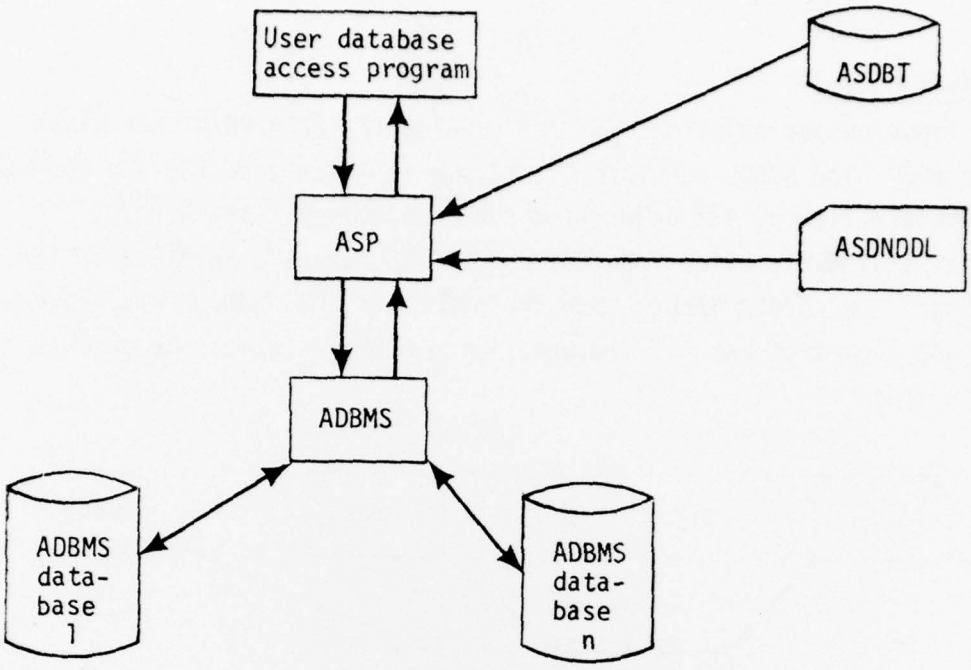


Figure 6-6
ASP Scenario

6.3.4 Components

The ASP consists basically of four modules, the ASDDLA, User Level Routines, Mid Level Routines, and Low Level, or Table Access Routines. The relationships among these modules are given in Figure 6-7.

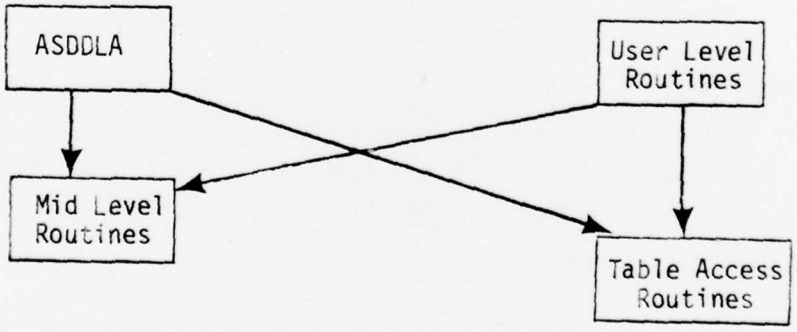


Figure 6-7

The User Level Routines, of which the ASDDLA is a subset, provide the user interface to the ASP. The Table Access Routines provide the interface between the ASP and the ASDBT. The intra-ASP relationships among the modules

and the relationships between the ASP modules and ADBMS are somewhat looser; accessing of the actual ADBMS databases takes places at all levels, with and without the aid of ADBMS.

6.3.4.1 ASDDLA Design

The ASDDLA is a compiler type analyzer consisting of the five components shown in Figure 6-8.

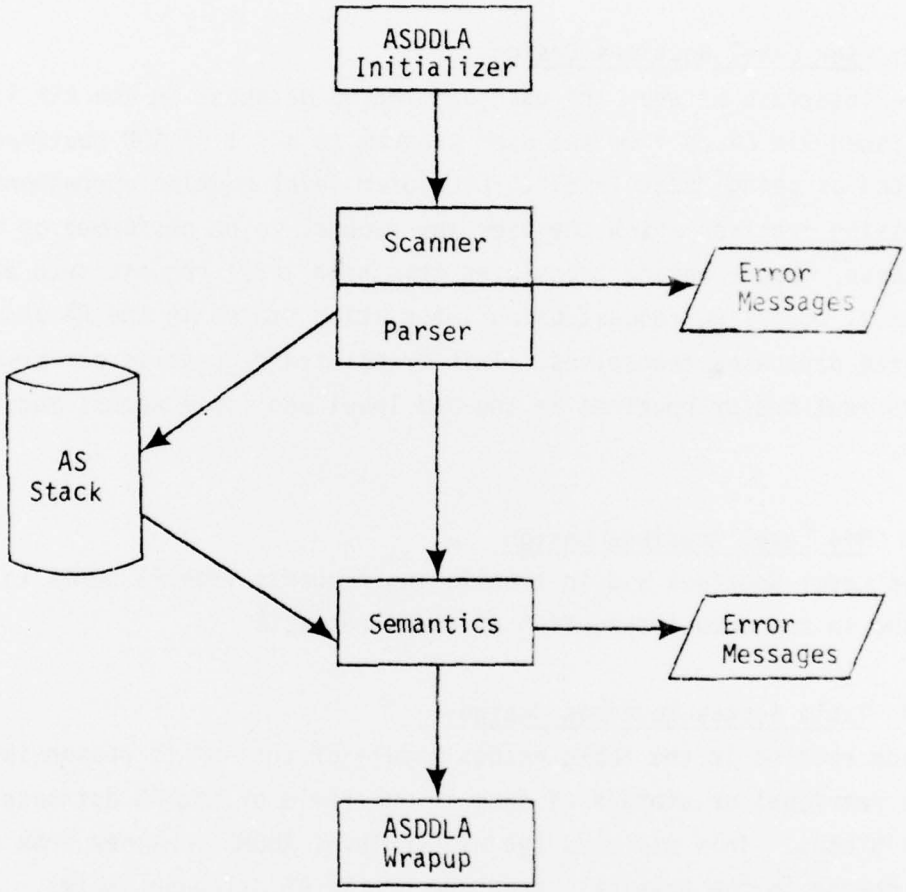


Figure 6-8
Aggregate Schema DDL Analyzer

The ASDDLA Initializer is the controlling program for all of the ASDDLA modules. It initializes arrays and prepares the scanner for input. The scanner and parser identify ASDDL constructs and resolve the ASDDL into various productions. Once a production has been isolated, the appropriate semantic routine is called. The semantic routines build the AS database tables and check for internal consistency. The ASDDLA wrapup module prints out a review of all table blocks generated and statistics for the analyzer run.

6.3.4.2 User Level Routines Design

The interface between the user and the AS database in the ASP is accomplished via CALLs from the user program to a set of ASP routines designated as being "user level". Each user level routine corresponds to an accessing function which the user may request to be performed on the AS database. Each routine translates this high level request into an ADBMS-level accessing request using information stored in the AS and underlying databases themselves. This translated request is performed by ADBMS routines or routines in the mid level and table access modules of the ASP.

6.3.4.3 Mid Level Routines Design

Mid Level Routines aid in translating requests from AS terms to ADBMS terms and in the actual execution of these requests.

6.3.4.4 Table Access Routines Design

Each routine in the table access module of the ASP is responsible for the retrieval or storage of data in one field of the AS database table control blocks. This isolates the higher level ADBMS routines from any design change in the physical structure of the AS database tables.

REFERENCES

(R) RESTRUCTURING

R2 NAVATHE, S. B. and FRY, J. P., "Restructuring for Large Data Bases," ACM Transactions on Database Systems 1,2 (June 1976), ACM, New York, 1976, pp. 138-158.

(UR) UM RESTRUCTURING

UR3 DEPPE, M. E., "A Relational Interface Model for Database Restructuring", Technical Report 76 DT 3, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1976.

UR5 DEPPE, M. E., and LEWIS, K. H., "Data Translation Definition Language Reference Manual for Version IIA Release 1", Working Paper 76 DT 5.2, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1976.

UR6 SWARTWOUT,D. E., MARINE, A. M., and BAKKOM, D. E., "Partial Restructuring Approach to Data Translation", Working Paper 76 DT 3.1, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1976.

UR7 SWARTWOUT, D. E., WOLFE, G. J., and BURPEE, C. E., "Translation Definition Language Reference Manual for Version IIA Translator, Release 3", Working Paper 77 DT 5.3, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1977.

(UT) UM TRANSLATION

UT1 "Functional Design Requirements for a Prototype Data Translator", Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1972.

UT2 "Design Specifications of a Prototype Data Translator", Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1972.

UT3 "Program Logic Manual for the University of Michigan Prototype Data Translator," Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1973.

UT4 "Users Manuals for the University of Michigan Prototype Data Translator:, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1973.

UT5 "Functional Design Requirements of the Version I Translator", Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1973.

UT6 "Program Logic Manual for the University of Micnigan Version I Data Translator", Working Paper 306, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1974.

UT7 "Design Specifications: Version II Data Translator", Working Paper 307, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1975.

UT8 BIRSS, E., DEPPE, M., and FRY, J., "Research and Data Reorganization Capabilities for the Version IIA Data Translator", Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1975.

UT9 BIRSS, E., et al., "Program Logic Manual for the Version IIA Data Translator", Working Paper 76 DT 3.1, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1976.

UT10 BODWIN, J., et al., "Data Translator Version IIA Release 1 User Manual", Working Paper 76 DT 3.2, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1976.

UT11 BODWIN, J., et al., "Data Translator Version IIA Release 2 User Manual", Working Paper 76 DT 3.4, Database Systems Research Group, The University of Michigan, Ann Arbor, Michigan, 1976.

UT12 KINTZER, E., et al., "Michigan Data Translator Version IIB Release 1.1 User Manual", Technical Paper 77 DT 8, Database Systems Research Group, The University of Michigan, Ann Arbor, Michigan, 1977.

UT13 BURPEE, C. E., et al., "Michigan Translator Program Logic Manual Version IIB, Release 1", Working Paper 77 DT 3.7, Database Systems Research Group, The University of Michigan, Ann Arbor, Michigan, 1977.